

Final Report

to

**General Motors Systems Engineering Center
Troy, MI**

**THE REQUIREMENTS FOR AN OBJECT ORIENTED
VEHICLE MODEL**

Contract # DD 492465

and

**HIERARCHICAL DECOMPOSITION DESIGN METHODS
FOR AUTOMOBILES**

Contract # DD 492386

J. Colton, J. Craig, G. Fadel, R. Fulton, J.E. Rogan

**W. Chang, D. Heifetz, J. Lambright, A. LeBlanc, W. Mayville,
J. Preminger, E. Stephens, P. Wijntjes**

**College of Engineering
Georgia Institute of Technology
Atlanta, GA 30332**

September 17, 1990

ABSTRACT.....	1
Chapter 1 INTRODUCTION.....	2
Chapter 2 OVERALL DESIGN STRATEGIES.....	4
2.1 Introduction	4
2.2 Issues in Product Development Decision-making.....	6
2.2.1 Vehicle Function and System Architectures	6
2.2.2 A View of the Automobile Design Process	8
2.2.2.1 Problems with the Current Product Development Process.....	8
2.2.2.2 The Impact of computing technology on the product development process.....	10
2.2.2.3 A System View of a Computing Environment for Concurrent Engineering.....	14
2.3 Basic Concept- A Multiview Design Approach for Automobiles	16
2.4 Research Approach.....	16
Chapter 3 CUSTOMER PERSPECTIVE OF THE VEHICLE.....	18
3.1 Customer Perspective via QFD	18
3.2 Baseline Parameters.....	18
Chapter 4 OPERATIONAL PERSPECTIVE OF THE VEHICLE.....	21
4.1 Introduction	21
4.2 Functional Description Development	21
4.2.1 Functional Decomposition.....	21
4.2.2 Function Relationships.....	23
4.3 Conceptual Level Application.....	39
4.3.1 Functional Description Subset.....	39
4.3.2 Function Relationships.....	41
4.4 Summary	41
Chapter 5 PHYSICAL PERSPECTIVE OF THE VEHICLE.....	44
5.1 Introduction	44
5.2 Vehicle Form Description.....	44
5.2.1 Input Parameters	44
5.2.2 Output Parameters	46
5.3 Vehicle Form Decomposition	49
5.3.1 Form Decomposition	49
5.3.2 Form Relationships.....	51
5.4 Summary.....	59
Chapter 6 A SAMPLE OF VOICE OF THE CUSTOMER/ FUNCTION/ FORM LINKAGES.....	62
6.1 Introduction	62
6.2 Engine Subsystem	62
6.3 Software.....	62
6.3.1 The User-interface	62
6.3.2 The VOC ->Functions Translator.....	63
6.3.3 The Function-optimizer.....	64
6.3.4 The Function -> Form Mapper.....	65
6.3.5 The Form-optimizer	68
6.4 Oracle Database	68
6.5 Conclusions.....	69

6.6	References.....	70
6.A.1	Definitions	71
6.A.2	Memory Problem using Macintosh's Think C and ORACLE.....	71
6.A.3	The Hard- and Software used	71
Chapter 7	A THEORETICAL APPROACH TO HIERARCHICAL DESIGN.....	72
7.1	Introduction	72
7.2	Multiobjective Optimization and Design.....	72
7.2.1	A New Technique Using Parameter Passing.....	73
7.2.2	Application to an Automobile Configuration Problem.....	74
7.2.3	Analysis of Design Methodologies	79
7.2.4	A Design Methodology to Support Requirements Tradeoffs.....	82
7.2.5	Hierarchical Design using Optimizing Sequences of Decisions	86
7.3	Organization of a Probabilistic Decision-making Strategy.....	93
7.4	Basic Concepts Underlying Taguchi and Tse Approaches to Design.....	94
7.5	References.....	100
7.A.1	Solution of Requirements Negotiation Problem using Methodology F.	101
Chapter 8	MULTIVIEW DESIGN APPLICATION.....	104
8.1	Introduction	104
8.4	Applications	104
8.4.1	Form-Function Test with Smalltalk	104
8.4.1.1	Future Considerations	107
8.4.2	Work Related to OOPM	108
8.4.3	Building the OOPM	109
8.4.4	Multiple Views	112
8.4.5	Example	112
8.4.6	Decomposition	115
8.5	Summary	119
8.6	References.....	119
Chapter 9	MULTILEVEL DESIGN APPLICATION	122
9.1	Description of the Automobile Door Design Example.....	122
9.2	Software Infrastructure	123
9.2.1	Software Platform	124
9.1.1.5	Description and Functions of Software Components	124
9.3	Multilevel Model of Door Geometry	126
9.3.1	Door Subsystem Modelled with ICAD	126
9.3.2	Detail Component Modelled with IDEAS.....	133
9.4	Summary	137
Chapter 10	SUMMARY OF ACCOMPLISHMENTS AND ISSUES.....	138
Chapter 11	PLANS	140
11.1	Planned Tasks.....	140

11.1.1 Task 1- Implement and Test Prototype Software Framework	140
11.1.2 Task 2- Refine and Extend Automotive Design Methodology	141
11.1.3 Task 3- Develop a Database Approach for Automotive Design	141
11.1.4 Task 4- Apply Prototype Software Framework to Representative Vehicle Design	142
11.2 Testing Methodology	142
Appendix A Quality Function Deployment	144
A.1 What is Quality Function Deployment?	144
A.2 Quality Function Deployment in Japan	144
A.3 Quality Function Deployment in the U.S.	145
A.4 The House of Quality	145
A.5 References	150
Appendix B FUNCTION DATA DICTIONARY	151
Appendix C FUNCTION AND DESIGN SPECIFICATION MAPPING	154
Appendix D OBJECT ORIENTED ANALYSIS AND DESIGN METHODS	169
D.1 Why Objects?	169
D.2 Background	169
D.2.1 History of Object Oriented Programming (OOP)	169
D.2.2 What Constitutes an OOP?	170
D.2.3 Definitions	170
D.2.4 Role of Relations	172
D.2.5 Selection	173
D.2.6 Data Storage in OOP	173
D.3 Existing Methods	174
D.3.1 Booch Method	174
D.3.2 Coad/Yourdon Method	175
D.3.3 OMT - Loomis, Rumbaugh, and Shah at GE/Calma	175
D.3.4 ObjectOry - Objective Systems AB	176
D.3.5 Shlaer/Mellor' Object Techniques	177
D.3.6 HOOD - European Space Agency	177
D.3.7 OOSD - Object Oriented Structured Design	178
D.3.8 OBA - Object Behavior Analysis	178
D.4 References	179

ABSTRACT

This report presents the results of the past year's work under contracts entitled The Requirements for an Object Oriented Vehicle Model (Contract # DD 492465) and Hierarchical Decomposition Design Methods for Automobiles (Contract # DD 492386) for the GM Systems Engineering Center, Troy, MI. The technical contacts are Dr. Steve Rohde and Dr. Dennis Li.

This research has provided the preliminary requirements for an object oriented vehicle model and the hierarchical decomposition strategies for maneuvering through the design of an automobile. The model provides an integrative focal point for all functions, so that they may each have their own view of the design without the problems associated with multiple views. Hierarchical design methodologies have been studied and conclusions made as to strategies to follow.

Chapter 1 INTRODUCTION

The design of a complex product, such as an automobile, requires the integration of the efforts of many different groups within a company. These can include design, manufacturing, analysis, project management, marketing, as well as a host of others. Each of these groups has its own design methods and, as a result, different views of what constitutes the vehicle and its descriptive information. This leads to redundancy and difficulty in communication, as groups compare different information perspectives, such as design and manufacturing descriptions. Each group has different types of input, output, and levels of detail. By consolidating the various models into one object-oriented representation of the vehicle with the ability to work at different levels of detail, and combining it with hierarchical decomposition strategies, a clear determination of a design procedure should emerge that incorporates design decisions and processes, multidisciplinary interactions and system optimization. This approach also should allow the formulation of an objective function for the conceptual design, with appropriate weighting functions to replace the current set of inconsistent parameters. This strategy should lead to a reduction in the time and cost of design, an improved vehicle quality, and increased market share.

In general, the design of an automotive vehicle should proceed from the desires of the customer (the voice of the customer, as developed within Quality Function Deployment (QFD)) to an engineering description and on to design, analysis and manufacturing. In the concurrent engineering view, the design of the product and the processes for making it should be performed at the same time. In a similar vein, design should be approached in a multilevel, multiview approach. The functional requirements of the vehicle should be determined from the customer requirements. These should then be translated into the forms required to produce these functions. The forms then need to be integrated to insure that the functions are performed to the necessary level. A model of the vehicle and the design process needs to be developed so that all of the functions involved can view the information according to their own needs, yet provide a consistent information base. An object oriented vehicle model will provide the integration necessary to accomplish this goal of providing a consistent information model that allows multiple views at multiple levels. Design methodologies, such as hierarchical decomposition strategies, will allow the designer to traverse the design in an optimal manner by pointing out the relationships between function and form, and the best way to perform the design.

This report summarizes the efforts carried out over the past year under two GM SEC contracts to utilize object oriented technology and hierarchical decomposition methodologies to provide a consistent, comprehensive and structured approach to the early design process for automotive vehicles. It is to take advantage of the concept of design being

an information-driven process and explore how new technologies in information management can aid the process. The first goal of the work summarized here is to develop an appropriate vehicle model which will integrate the multiple levels of the vehicle as well as allow each function to view the vehicle on its own terms. The second goal is to provide an integrative, optimal design strategy and process, which take into consideration the interactions of the voice of the customer with the function and form of the vehicle, via an optimizable functional description. The approach will utilize hierarchical decomposition design methods, which will provide an ordered way of addressing various levels of design definition and decisions. Such a structured approach should result in a reduction in feedback within the design process and an associated reduction in design time.

Work in the past year has determined the preliminary, theoretical requirements for such an Object Oriented Vehicle Model (OOVM) and Hierarchical Decomposition Design Methods for Automobiles. It is felt that the OOVM will provide a consistent, integrative representation of a vehicle for use by all functions involved in the design of automobiles and that the Hierarchical Decomposition Methods will provide optimal strategies for the design of the vehicle.

Chapter two presents an overview of the design process and our overall design strategies. Chapter three discusses the customer perspective of the vehicle as it is related to QFD and selected Baseline Parameters. Chapter four discusses the functional decomposition of the OOVM and chapter five discusses its form decomposition. Chapter six presents a preliminary integration of the OOVM with a database and an expert system to test the linkages between function and form. Chapter seven presents the theoretical approach to hierarchical design and optimization strategies. Chapter eight discusses the object oriented programming paradigm to obtain multiple views of the design with applications. Chapter nine presents a similar discourse on multiple level views of the design, again with an example. Chapter 10 summarizes the results of this work and chapter 11 presents our plans for the future.

Chapter 2 OVERALL DESIGN STRATEGIES

2.1 Introduction

The design of an automobile is a complex process of synthesis, analysis and decision-making. The vehicle development process involves interactions among many different disciplines and spans a chronology from the earliest expression of the voice of the customer, through the design and manufacture of tens of thousands of vehicles, and finally to the operation and support of a large and widely distributed fleet. The design and development of complex automotive vehicles requires the consideration of numerous disciplines, complex geometries, materials and processes, the involvement of numerous specialists, and extensive task automation. Consideration of the entire vehicle life-cycle from the earliest phases of the project is a key element of success in a highly competitive and rapidly changing marketplace.

Design processes for future vehicles require a high degree of automation, integrated into a cohesive product development environment. Current approaches utilize computers in varied ways to automate tasks, but do not generally address the entire process in a coherent manner. To achieve product quality and development productivity, it is essential that methods be developed to manage design information and to structure the decision-making process throughout the vehicle development process. An approach that appears to have considerable merit is one based on hierarchical decomposition of the critical design tasks and variables into appropriate groups consistent with the state of the design.

The present research program in hierarchical decomposition methods and vehicle modelling is built around three inter-related efforts that are being carried out concurrently by the research team. These efforts are focused on

- data modelling for Quality Function Deployment (QFD),
- development of an object-oriented vehicle model, and
- methods for making product development decisions.

In this approach, QFD provides a problem statement for decision-making, which will be discussed more completely in chapter three and appendix A.

An object-oriented vehicle model provides a description of alternative vehicle system, subsystem, and component attributes. The initial focus for the OOVMM is conceptual design (i.e., in which major vehicle characteristics such as length, width, and power characteristics are determined) to examine the applicability and link of the model to detailed design (i.e., specification of bolt sizes, body panel thickness, etc.). The basic concept is to identify the key design variables in conceptual vehicle design

and the steps required to assign values to those design variables. We also want to organize these steps in the optimal order so as to make design "feedforward" and minimize the need for feedback and iteration, thereby minimizing both the complexity of the design process and the time needed for design. Expressed another way, we must determine what design decisions have to be made and how and in what order those decisions should be made to optimize conceptual vehicle design.

Of course, the goal of any design is to assign values to the design variables so that the product meets the requirements of the person (or organization) that requested the product be designed (and also satisfies any constraints). In our case, this person is the vehicle customer. It is our contention that through a progressive process of analyzing and refining the functional requirements of the customer (i.e., what the customer wants the car "to do"), selecting and designing forms to fulfill those functional requirements, then synthesizing (and optimizing, where necessary) those forms into an overall vehicle, the OOVm will converge upon an optimal solution (i.e., a vehicle design that optimally meets the customer requirements). Specifically, the input to the model is the "voice of the customer" (or customer attributes). These (or, more exactly, values assigned to these attributes by the customer) are translated into functional requirements, and in turn, the functional requirements are translated into forms. The forms are then optimized (under the requirements of functional and spatial constraints) and aggregated into an overall vehicle. The output of the model is the final form and functional characteristics (or product characteristics) of the "optimal" vehicle design. Our ultimate goal is to show how, as the input customer attributes change, the output product characteristics change.

In general, the decision-making task is to select values for the attributes, which satisfies hard constraints and balance goals and objectives. The present effort is concerned with the development and prototype implementation of such decision-making methods. In particular, the present approach is focused on the development of decision-making strategies in a hierarchical design environment. Multilevel optimization methods, deterministic decision-support processes, and probabilistic methods, such as Taguchi methods and related developments, are appropriate for making various design decisions and are currently within the scope of this study.

A key idea in this approach is that the prioritization of goals, objectives, and conflicting requirements is incorporated into the product through the strategy for converging on final specifications for product attributes. Hierarchical design methods can play two roles:

1. in planning a sequential or concurrent product development decision-making process to ensure that requirements are balanced, and
2. in providing support in making those decisions.

A principal tool in hierarchical design approaches has been the iterative solution of an optimization problem decomposed into a multi-level network of subproblems. The use of the word hierarchical refers to the multi-level nature of this network. The techniques can be applied to quite general problem structure networks, provided some care is taken to avoid divergence. This is important, since the techniques are applied in practice to what might be described as the "decision decomposition". The decision decomposition rarely has a strictly hierarchical structure, since it must reflect both the function and system hierarchy decompositions. (The function decomposition is typically not hierarchical.)

The primary emphasis in the development of these tools has been in solving problems, which could be posed as deterministic single-objective multivariable constrained minimization problems. The techniques are thus somewhat limited in their ability to deal with independent variables taking on values in a discrete set, or to deal with objective and constraint functions having discontinuities. Also, the set of independent variables, objectives, and constraint relationships is generally taken as fixed. This limits the ability of these methods to deal with qualitatively different alternatives. Such alternatives are characterized by differences in system architecture. Architectural differences, in this sense, involve different function decompositions, and different choices of system elements for implementation. Finally, the "classical" approach to hierarchical design decision-making provides little guidance in dealing with formulation or solution of problems where uncertainty or imprecision are important.

As a result of these issues, the present study also examines alternative decision-making methods based on both deterministic and probabilistic formulations. The probabilistic formulation brings with it a strategy for integrating concurrent decisions. The result is a broad-based approach to the development of decision-making strategies that are appropriate to the problem of designing complex mechanical systems, such as an automobile.

2.2 Issues in Product Development Decision-making

2.2.1 Vehicle Function and System Architectures

In the approach taken here, hierarchical design methods fit into the context of an advanced product development methodology, based on the concept of concurrent engineering. Concurrent engineering refers to a product development process in which producibility and supportability considerations can be brought into early product decision-making steps and traded off against performance, cost, and schedule (time-to-market).

A critique of the product development process, as it is often now conceptualized, is presented. It is shown that certain obstacles to

concurrent engineering are built into this view of the product development process. Computing technology can be applied to allow substantial changes in the structure of the product development process. In particular, high-level languages for parametric representation of product and process alternatives are extremely valuable.

To be effective, these languages must allow the product development team to describe the system life cycle concept from three points of view:

- function ("what it does"),
- implementation ("what it is") and
- modelling ("how it works").

Functions are organized by a function decomposition structure, while implementation alternatives can be organized into various system-system-component hierarchies. Both functions and system elements have attributes. Customer, regulatory, and internal requirements can be applied to the life-cycle concept description by constraining the values of these attributes. Goals and objectives, applied to these attributes, constitute additional constraints (optimality constraints). Finally, engineering theories and models ("how it works") link attributes of the function and system hierarchy decompositions. These models provide still more constraints. The decision-making task is to select values for the attributes, satisfying the hard constraints, and balancing goals and objectives.

A key idea in the current approach at Georgia Tech is that the customer's prioritization of goals, objectives, and conflicting requirements is incorporated into the product through the strategy employed by the product development team to converge on final specifications for product attributes. Hierarchical design methods can play two roles: (1) in planning a sequence of product development decisions to ensure that requirements are balanced, and (2) providing support in making those decisions.

The principal tool in the hierarchical design approach has been the iterative solution of an optimization problem decomposed into a multi-level network of subproblems. The use of the word hierarchical refers to the multi-level nature of this network. The techniques can be applied to quite general problem structure networks, provided some care is taken to avoid divergence. This is important, since the techniques are applied in practice to what might be described as the "decision decomposition". The decision decomposition rarely has a strictly hierarchical structure, since it must reflect both the function and system hierarchy decompositions.

The primary emphasis in the development of these tools has been in solving problems which are posed as deterministic single- or multi-objective multivariable constrained minimization problems. The techniques that have been developed are thus somewhat limited in their ability to deal with independent variables taking on values in a discrete set, or to deal with objective and constraint functions having discontinuities.

Also, the set of independent variables, objectives, and constraint relationships is generally taken as fixed. This limits the ability of these methods to deal with qualitatively different alternatives. Such alternatives are characterized by differences in system architecture. Architectural differences, in this sense, involve different function decompositions, and different choices of system elements for implementation. Finally, the "classical" approach to hierarchical design decision making provides little guidance in dealing with formulation or solution of problems where uncertainty or imprecision are important.

2.2.2 A View of the Automobile Design Process

In the following sections, a brief description of the product development process as it is widely implemented today is presented. The attempt here is not to try and accurately represent a development process actually in use by GM, but rather to describe general features of a "traditional" product development process. This description then provides the material for a critical analysis and the basis for formulation of proposed improvements based on hierarchical design methods.

2.2.2.1 Problems with the Current Product Development Process

One view of the current product development process is shown in Figure 2.1 (numbers in parentheses refer to points in the figure). In this approach, requirements are defined first. Next, (1), an initial specification of the product is based on these requirements. An example is a drawing or three-dimensional CAD model of a part. Since it is not known *a priori* whether the product specification meets the requirements, specialists in producibility, supportability, and other disciplines participate in a design review, (2). Typically, problems are identified with the initial product specification, and solutions are proposed. These changes are immediately made, (3), if cost and schedule considerations allow modification of the design, and other requirements need not be compromised. If compromise is required, a conflict resolution process is invoked, (4-5), typically through the management structure of the product development organization. This may result in changes to the product specification, (6), or modification of the requirements, (7). If the conflict resolution process converges on a satisfactory design, the product specification is released for production and support planning, (8).

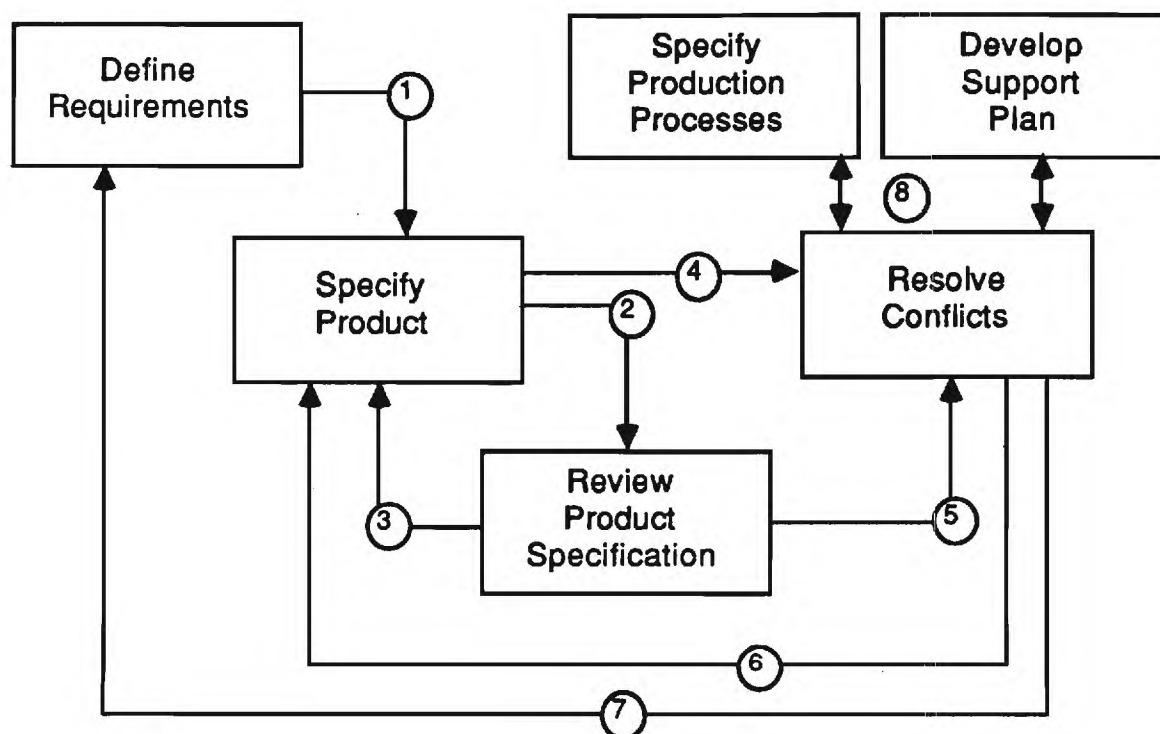


Figure 2.1. Existing Product Development Process.

Obstacles to concurrent engineering are built-in to the existing product development process. For product development projects involving compromise, the system life cycle concept emerging from the product development process represents a prioritization of the requirements. This prioritization is reflected in the sequence in which product, process, and planning decisions are made. The sequence of decisions is particularly important, since each decision may place severe constraints on the options available for subsequent decisions.

This activity, matching decision-making priorities for product and process specifications to customer requirements, is remarkably absent from the structure of the current product development process. In the current process, we rely on the experience of the designer and the effectiveness of the design review process, especially the conflict resolution abilities of the technical project leader, to ensure that requirements are being met.

In fact, using existing design technologies, the flexibility to make design changes that is available to the designer and the project leader is often limited. The cost of generating or modifying product specifications may make it impossible to make the necessary design changes, or there may not be time to make them. In practice, decisions are sequenced to meet the schedule for releasing product specification information. These

schedules are prepared before the product specification and design review processes are initiated. Thus, it is rarely possible to ensure that the decision-making priorities implicit in the product specification release schedule accurately reflect the impact of these decisions on customer requirements.

In the current product development process, design review for producibility and supportability is decoupled from planning of the production process and support operations. The separation of design review from production and support planning has been necessitated by the cost of production planning, and the need for highly detailed product specification information before production and support planning could be initiated. As a consequence of this separation, information is often lost in the transition. In fact, decisions are made during the product specification and design review process that severely restrict the options left for production and support planning. The full implications of these decisions may not be evident until planning for production and support is started.

2.2.2.2 The Impact of Computing Technology on the Product Development Process

Instantiation, constraint propagation, and the services required to distribute computational objects to multiple physical locations and to maintain versions of them over time are the computing technologies which appear to have the most powerful impact on the product development process. The most fundamental change has been in the application of the object-oriented programming style and constraint propagation to develop high-level languages to support parametric description of product and process alternatives. The next step is clearly the development of distributed objectbase capability which is one of the key elements needed to "scale up" parametric product/process description technology for production applications in industry.

A common initial impression is that parametric product definition technology impacts only the product specification aspects of the development process. In fact, parametric description makes it economically feasible to effect fundamental changes in the structure of the entire product development decision-making process. This is a result of the fact that parametric description technology changes the economics of product specification.

Computer programs for parametric description can be used to generate detailed product specifications, based on a description of the design in terms of system integration parameters. Such computer programs have been in use for many years. What has changed, quite recently, is that the object-centered programming style has been exploited to significantly lower the number of engineering hours required to write computer programs for parametric description. Using a high-level

language to model elements of the product, new computer programs for parametric description can be created for roughly the same cost (in engineering hours) as a three-dimensional CAD model. The difference, of course, is that if a parameter value must be changed during the system integration process, the cost of changing a conventional CAD model is practically the same as the cost of developing the original product specification in CAD. Since details of a design that has been defined parametrically can be changed by simply executing the parametric description procedure with new values for some of the parameters, the engineering cost of making a change in the product specification is negligible.

Of course, this technology can be applied to execute the iterative loops in the current product development process more quickly. However, unless producibility and supportability can be incorporated into the procedures on which the parametric description computer programs are based, the increase in the speed with which design detail can be generated may actually heighten the obstacles to concurrent engineering. Providing designers with the ability to more rapidly generate product specifications with poor producibility and supportability characteristics is likely to overwhelm the design review process.

Moreover, parametric description, by itself, contributes nothing to the solution of the problem of balancing "downstream" producibility and supportability considerations against performance and product specification schedule and cost. This can only be changed by modifying the sequence in which product development decisions are made to better match the customer's prioritization of requirements. It is important to note, however, that parametric description technology provides the designer and the project leader with a considerable enhancement in their flexibility to sequence decisions. Parametric description technology frees the process from the constraints imposed by the cost of considering several alternative concepts for the product, as long as they can be effectively modelled using the parametric description language.

There are thus three modifications that must be made in the structure of the design process, in order to use parametric description technology to accomplish some of the aims of concurrent engineering:

1. the "specify product" step of the existing process must be replaced by a step in the "to-be" product development process in which alternative product, manufacturing process, and support concepts are parametrically modelled. The entire concurrent engineering team must be involved in the development of computer programs for parametric description.

2. an explicit planning step must be included to ensure that the sequence of product development decisions reflects the prioritization of conflicting requirements by the customer. Again, the entire concurrent engineering team must be involved.
3. The decision-making process must balance the views of all members of the concurrent engineering team.

The effect of these changes on the structure of the product development process is indicated in Figure 2.2.

Computing technology, through application to parametric description, can play a key role in making these structural changes in the product development process possible. In order to realize these gains, problems of "scaling" - providing a distributed environment to support the modified product development process - must be solved. Techniques for planning the decision-making process must also be further developed and demonstrated.

There is one other aspect of the impact of computing technology on the product development process which is important in the proposed approach to hierarchical design methods. This has to do with the development of systems. Systems are now developed hierarchically. In a hierarchical approach, product specifications are defined in stages. Increasing levels of detail are specified at each stage. This means that the product is only partially specified when decisions are made.

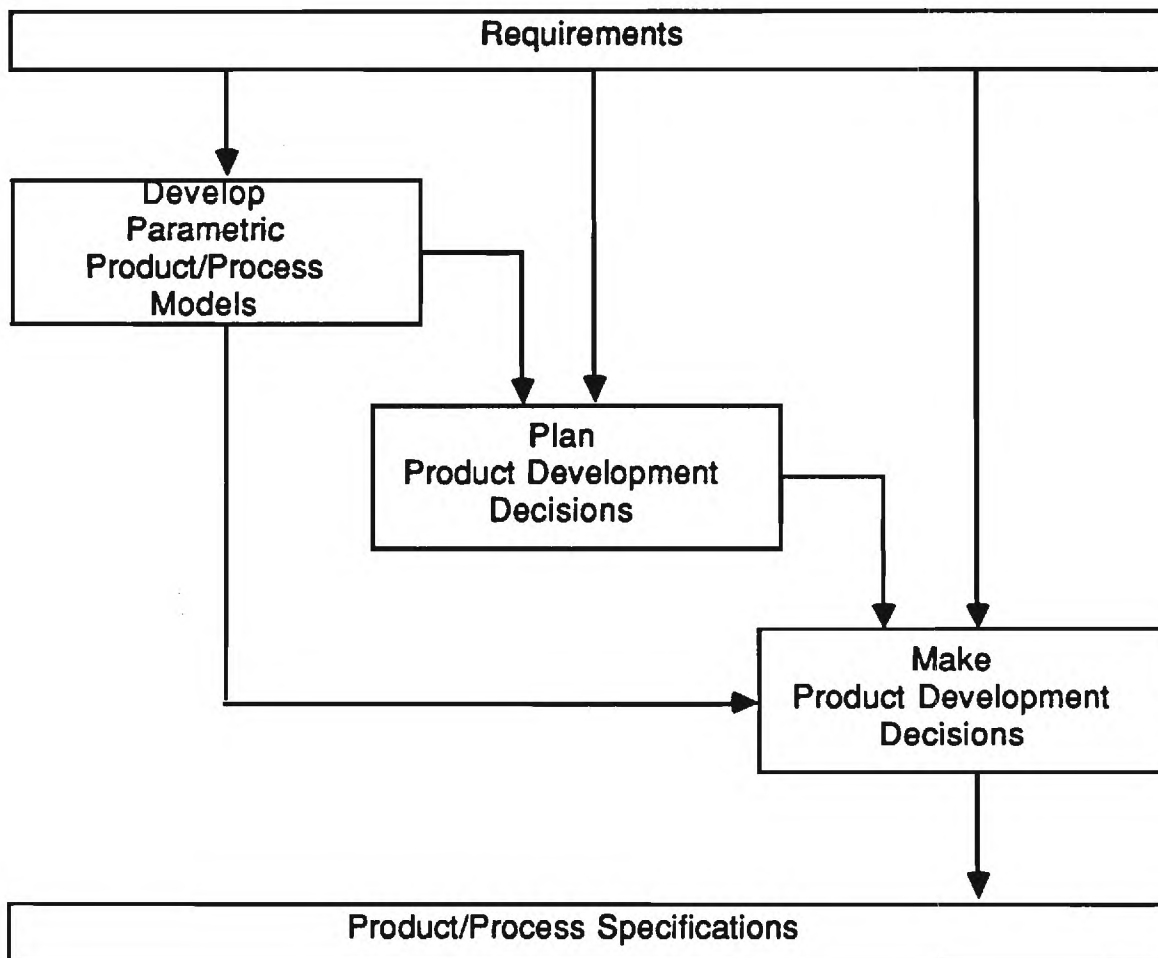


Figure 2.2. A Product Development Process for Concurrent Engineering.

In principle, it should be possible to use parametric description technology to fully detail the design before making any decisions. This is a consequence of the fact that the cost of making changes to the product specification as alternative concepts are considered is extremely low. This approach may, in fact, be practical for some types of products which are derivatives of similar systems.

For systems incorporating new technologies or inventions, the cost of creating the parametric models for elements of the product and manufacturing process is a significant portion of the product development investment. In addition, modelling of advanced technology components often involves significantly higher levels of technical risk. These factors in combination necessitate the use of a hierarchical approach, even in a parametric description environment. Thus, we must address the question of making decisions based on a partial specification of the product. Techniques for attacking this question are highly applicable to the

development of hierarchical design methods, and form a cornerstone of our technical approach.

2.2.2.3 A System View of a Computing Environment for Concurrent Engineering

A successful architecture for concurrent engineering (CE) must support several possible CE computing environments. One type of CE environment will be presented here. In this particular environment, high-level product and process modelling languages are used to create parametric models. The integrating technology is a distributed objectbase containing models of the system as well as objects which are used to manage the state of the development process. Elements of such an environment are indicated in Figure 2.3.

A distinction between functional architecture ("what it does" or "how it is used") and implementation architecture ("how do we build it" or "what it is") must be made. Elements of a functional architecture are shown in Figure 2.3. In this section of the report, the concern is almost exclusively with the functional architecture. Thus, when the word CE architecture is used without qualification, the reference is to the functional architecture.

Specification of an architecture includes both the decomposition of the system into elements and a description of the interfaces between those elements. First, it is necessary to identify and describe the elements of the architecture of Figure 2.3. The architecture in Figure 2.3 decomposes a CE environment into two subsystems: team interfaces and a distributed objectbase. A team interface supports each of the steps of the CE product development decision-making process shown in Figure 2.2. The development of parametric product/process models is supported by a high-level language interface. Similarly, interfaces are provided for planning and executing the product development decision-making process.

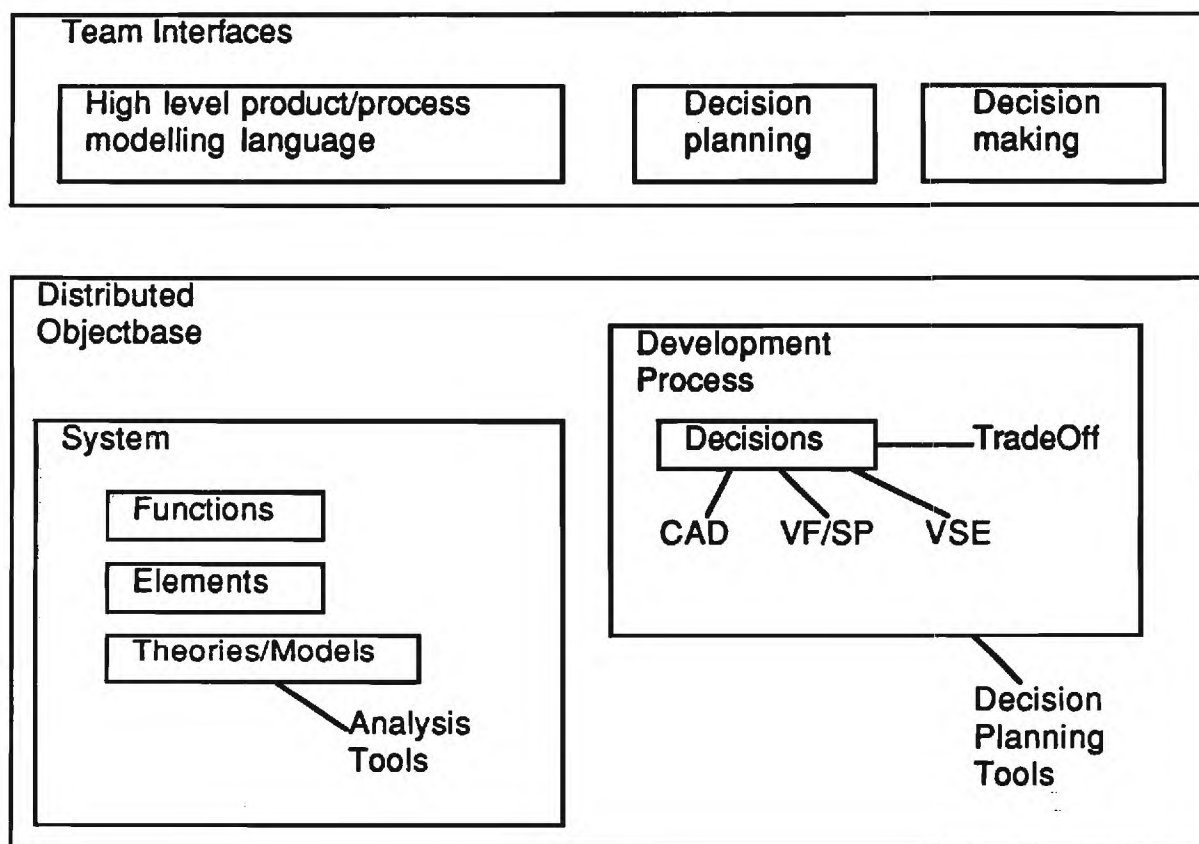


Figure 2.3. A System View of a Computing Environment for Concurrent Engineering.

The parametric descriptions of the system, developed using the high-level modelling language interface, are captured in the "system" element of the distributed objectbase. In order to support many CE objectives, the high-level modelling language must provide for the description of system functions ("what it does"), elements ("what it is") and theories and models. The theories and models describe how specific system elements work together to perform specific functions. For product development in response to complex requirements, or incorporating risky technologies, multiple levels of theories and models are often used to support product development decisions. Thus, these aspects of the system life cycle concept must be represented explicitly in the objectbase. Theory/model objects manage the execution of analysis tools as methods. This relationship is indicated in Figure 2.3 by a line connecting "Analysis Tools" to the box around the "Theories/Models" object class.

The parametric nature of the system description implies that, until product development decisions are actually made, the system description contained in the objectbase represents a catalog of alternative choices for product and process specifications.

Product development decision-making tasks are also represented explicitly in the distributed objectbase. These decisions are instantiated by the CE team during product development decision planning step. While the decision planning step is of considerable relevance to issues of hierarchical design methods, it is necessary to postpone detailed discussion of this step to the presentation of the automobile door design example in chapter 9. In this section, an outline of the barest elements of the process is presented.

Associated with each decision is a set of product and process attributes to be specified by execution of the decision-making step. The CE team identify these attributes, the required analyses, simulations, and tests, and the team members participating in the decision as part of the decision planning process. The CE team also develop a convergence strategy for these decisions in such a way as to properly balance customer requirements.

2.3 Basic Concept- A Multiview Design Approach for Automobiles

There are many different views of vehicle and each has a hierarchy. Each of these views has benefits for some applications or perspectives. The three views we have chosen include the customer perspective, the operational perspective, and the physical perspective. These views are linked together by mapping transformations. The key aspect of our work is (1) to develop an approach that allows the customer perspective to drive the design and (2) to allow each perspective to be continually refined to lower levels of definition until the design is completed. We are particularly interested in the early conceptual design phase where the design concept is set, but we also want to be able to refine the design at least through preliminary design. We are concerned with understanding and characterizing the above three views, as well as building links between the views.

2.4 Research Approach

The research approach in these two projects can be summarized in the following list:

1. Investigate each of the three perspectives.
2. Identify the top level characteristics and parameters.
3. Refine the perspectives hierarchically to further levels.
4. Build some representative linkages among the perspectives (views).
5. Develop representative databases containing the views.

6. Develop prototype software approaches for supporting this design approach.
7. Apply software to selected simple test situations to clarify issues and refine concepts.
 - Gross vehicle characterizations
 - Hierarchical design of component, subcomponent, and part
 - Optimization strategies
 - Trade-off study concepts
8. Establish a baseline approach for application to vehicle design problems.

The next several chapters provide the details for each of these areas.

Chapter 3 CUSTOMER PERSPECTIVE OF THE VEHICLE

This chapter discusses the customer perspective of the vehicle in terms of the voice of the customer through the Quality Function Deployment (QFD) methodology.

3.1 Customer Perspective via QFD

The customer's wants can be determined through the use of a QFD methodology, which is discussed in further detail in Appendix A. The results are known as the "voice of the customer" (VOC). This can be a quite detailed list of desires, such as roominess, acceleration, and gas mileage. Actual values can also be determined with this methodology. An extensive list is being developed by Wei Chang at the GM Systems Engineering Center. Therefore, we will not dwell on the results of his work. As this is a lengthy list, we have chosen to assume that it exists and continue our work assuming that its output values will serve as the input to the work being performed in this study. The linkages between the QFD and our system will be through the baseline parameters, discussed in this chapter and the inputs discussed in chapter five. The VOC will serve as the starting point of the design process resulting in a "customer-driven" design.

3.2 Baseline Parameters

The baseline parameters form a set of parameters that can be used to completely define an object that can be identified as an "automobile". They have been used to determine the input and output of the object oriented vehicle model. The context of this research is the "conceptual" level, that is, design of the major characteristics of a "car" (versus development of a new mode of transportation). Accordingly, we developed a set of baseline parameters which compose nine characteristics whose values can be used to distinguish between instances of vehicles at the conceptual level. These are follows:

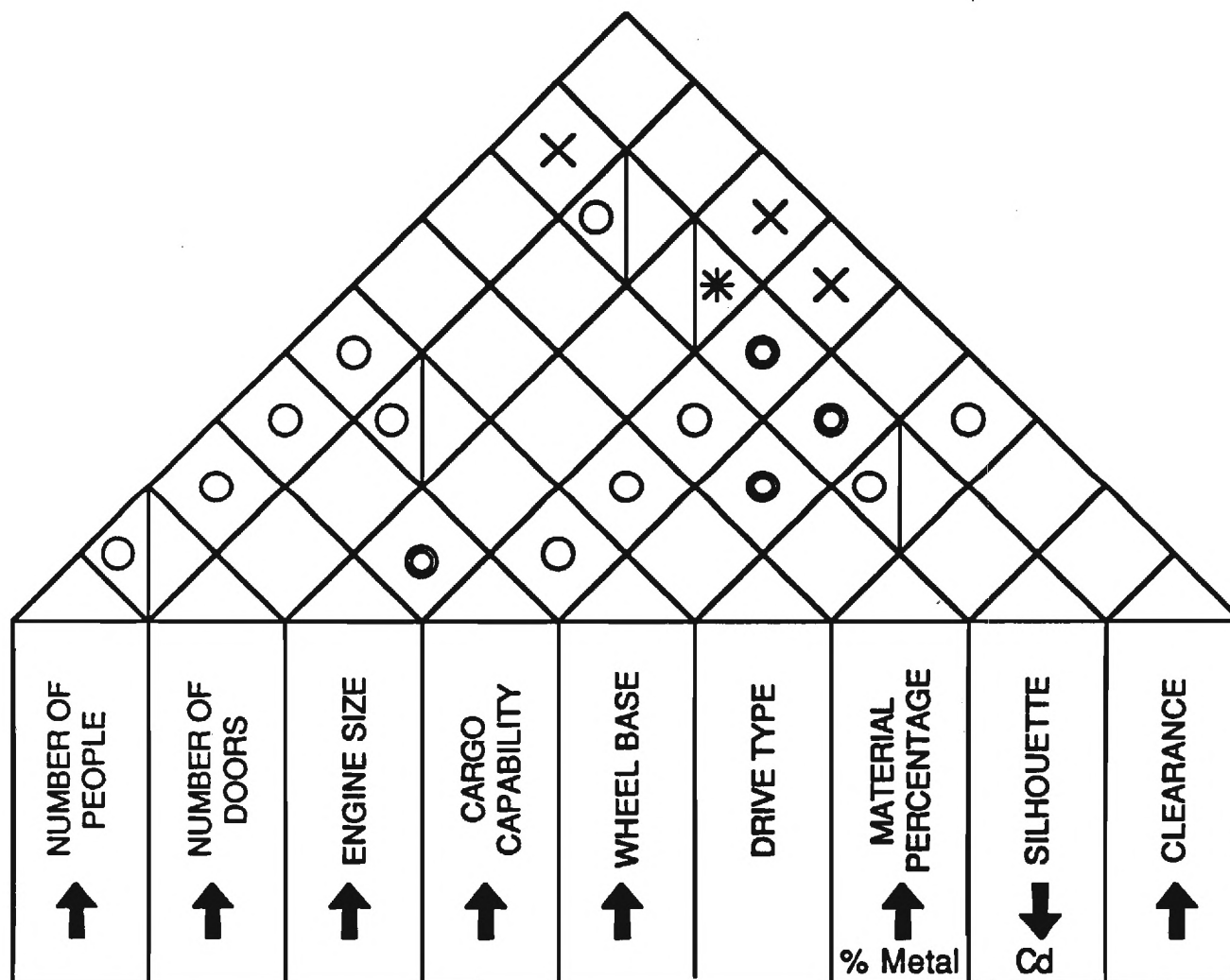
1. Number of People
2. Number of doors
3. Engine Size
4. Cargo Capability
5. Wheel base
6. Drive type
7. Material Percentage (% Metal)
8. Silhouette
9. Clearance

The interactions among the baseline parameters were explored through the use of a Quality Function Deployment (QFD) type of "House of Quality" (see Figure 3.1). The QFD house provides a simple, effective means by which to investigate and document qualitative relationships among variables. It also provides an easy to comprehend, graphical way to present these findings for review. The "roof" relationship matrix of the house was filled in using symbols to express the strength (strong or weak) and type (positive, negative, or none) of the interactions between the baseline parameters, which were arrived at through group consensus. The symbols reflect the type and degree of interaction that results from changing a value of a baseline parameter, and the arrows beneath the parameters indicate the direction of change. For example, if the engine size is increased, it will have a strong positive effect on the effort to increase the cargo capability (weightwise) of the vehicle. Note that the baseline parameter "drive type" does not have an arrow to indicate a direction of change. This is because a change of drive type amounts to a change in type (rather than change in value), i.e., to front, rear, or all wheel drive.

Interestingly, we found the need to modify the QFD house roof relationship matrix to provide indications of "directionality," because some of the interactions are directional. Directionality is shown by splitting a roof relationship "box" in two. When reading the relationships in the roof, the symbol that is seen first when proceeding from one parameter to the next is the one that should be used. For example, increasing the number of people the vehicle can transport has a weak positive relationship with increasing the number of doors (e.g., if it is desired to be able to accommodate 8 passengers, this will most likely require more than two doors), but increasing the number of doors doesn't necessarily have any effect on the number of people the vehicle can carry.

The baseline parameters form a link between the voice of the customer and the inputs to the functional description of the vehicle discussed in the next chapter.

BASELINE PARAMETER INTERACTIONS



© STRONG POSITIVE

○ WEAK POSITIVE

X WEAK NEGATIVE

*** STRONG NEGATIVE**

Figure 3.1

Chapter 4 OPERATIONAL PERSPECTIVE OF THE VEHICLE

4.1 Introduction

Functional description plays an important role in generating the information needed in the object oriented vehicle model. The functional description of a design specifies what the design must do, i.e., how it must function. This description type leads to improved designs by minimizing the information associated with the design. This minimization provides a broad yet restricted domain of operation for the design. The restrictions hold the design to what it must do to accomplish its goal. This eliminates redundant work and work not related to fulfilling a design function. A functional description allows for domain breadth by not describing how the design is to be accomplished, but allows for innovative combinations of function principles to attain the design goal. This chapter discusses the achievements in applying the functional description data to date and the plans for the future.

4.2 Functional Description Development

4.2.1 Functional Decomposition

The definition of a vehicle was established to guide the functional decomposition development. This was accomplished by defining what a vehicle is, as well as previous work at GM and Georgia Tech. The definition used was as follows:

Vehicle- A personal system of safe and comfortable transportation.

This definition was chosen because it defined what a car does without reference to what it is physically. This distinction is a purpose of functional description.

A vehicle is defined as having three overall functions, Transport, Please and Protect. Operate Safely forms an overall goal of any design. It is not included specifically as a function because all products designed should operate in a safe manner.

The hierarchical decomposition method used is based on the IDEF0 authoring process. This involves determining the possible subfunctions of a particular function and then examining for logical groupings of these activities. This was accomplished by asking "What does this function do?" The groupings were given a functional name. An example of a logical grouping is combining Start Energy Conversion and Stop Energy Conversion into the function Control Energy Conversion. These groupings were studied to ensure there was no overlap in the subactivities. It is important to note that between the three functions, Transport, Please and

Protect, there are overlaps or directly corresponding subfunctions. For instance, the function Accelerate Vehicle has a role in Transport, as well as Allow for Exhilaration. These functions are defined in the function data dictionary in Appendix B, in addition to the other functions. The vehicle functional description consists of three levels. Upon examination, the functions were found either to be performed directly by a passenger or driver, or enabled the passenger or driver to perform a function. The third level functions are categorized as either enabling or direct functions, or as both. The following are the functions specified for a vehicle:

<u>FUNCTION</u>	<u>TYPE</u>
F1 Transport	
F1.1 Provide Energy	
F1.1.1 Generate Useful Energy	Enable
F1.1.2 Distribute Energy	Enable
F1.2 Control Motion	
F1.2.1 Monitor State of Motion	Direct
F1.2.2 Control Energy Conversion	Direct
F1.2.3 Choose Direction	Direct
F1.2.4 Steer Vehicle	Direct
F1.2.5 Accelerate Vehicle	Direct
F1.2.6 Decelerate Vehicle	Direct
F1.3 Provide Space and Support	
F1.3.1 Provide Passenger Space	Enable
F1.3.2 Provide Cargo Space	Enable
F1.3.3 Provide Engine Space	Enable
F1.4 Access Vehicle	
F1.4.1 Access Passenger Space	Direct
F1.4.2 Access Cargo Space	Direct
F1.4.3 Access Engine Space	Direct
F1.5 Maintain Vehicle's Ability	
F1.5.1 Repair Malfunctions	Direct
F1.5.2 Perform Routine Maintenance	Direct
F2 Protect	
F2.1 Provide Safety	
F2.1.1 Prevent Mishaps	Both
F2.1.2 Mitigate Mishaps	Both
F2.2 Provide Security	
F2.2.1 Prevent Undesirable Intrusion	Both
F2.2.2 Prevent Vehicle Theft	Both

F3 Please**F3.1 Provide Physical Comfort**

F3.1.1 Maintain Ambient Temperature	Direct
F3.1.2 Provide Comfortable Spacing	Enable
F3.1.3 Provide Ergonomic Interfaces	Enable
F3.1.4 Provide Comfortable View	Enable
F3.1.5 Provide Comfortable Air	Enable
F3.1.6 Provide Comfortable Ride	Both
F3.1.7 Provide Low Noise	Enable

F3.2 Provide Mental Comfort

F3.2.1 Perform As Expected	Enable
F3.2.2 Provide Feeling of Value	Enable
F3.2.3 Provide Appropriate Look	Enable

F3.3 Entertain

F3.3.1 Allow for Exhilaration	Both
F3.3.2 Allow for Stimulating Environment	Both

NOTE: The function designations, such as F1 and F1.1, are read as Function 1 and Subfunction 1 of Function 1, respectively.

4.2.2 Function Relationships

Various relationships were mapped to assist the functional decomposition process. These relationships tested the completeness of the decomposition and provided other information. The functions were mapped to a list of GM functions developed in the GM report Vehicle System Description: A Preliminary Study, the Baseline Parameters, and the other functions. The last two sets of relationships are modeled by adapting the body of the "House of Quality", and the roof type mapping from the QFD. This is done because the QFD is an easy, graphical means for understanding relationships.

The functional description was mapped to previous GM work to test the completeness and quality of research work and to foster common terminology. This mapping was done for both the second level and third level functions. The second level map was used to guide the development of the third level map and ensure continuity within the hierarchy. The second level map was used as a guide and not a restriction because the mapping of the third level creates new information to be considered at all levels.

Figures 4.1 through 4.5 depict the function mappings to the second level functional decomposition. Figure 4.1 depicts the relationship of the GM function Transportation to the subfunctions of F1 Transport. For Figure 4.2, the correspondence is between the GM function Safety and F2 Protect. The mapping of GM Comfort versus F3 Please is in Figure 4.3. Also, F3 Please is mapped with the GM function Enjoyment in Figure 4.4. Figure 4.5 depicts the correspondence of the Energy and Information

GM TRANSPORTATION/GT TRANSPORT FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		TRANSPORT				
		DEVELOP MOTION	CONTROL MOTION	PROVIDE SPACE AND SUPPORT	ACCESS VEHICLE	MAINTAIN VEHICLE'S ABILITY
TRANSPORTATION	START CONTROL	✓	✓			
	SHUT OFF CONTROL	✓	✓			
	DIRECTIONAL SELECTION CONTROL		✓			
	ACCELERATION CONTROL	✓	✓			
	BRAKING CONTROL	✓	✓			
	STEERING AND HANDLING CONTROL		✓			

Figure 4.1

GM SAFETY/GT PROTECT FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		PROTECT	
		PROVIDE SAFETY	PROVIDE SECURITY
SAFETY	ACCIDENT AVOIDANCE	✓	
	HAZARD PROTECTION	✓	✓
	SECURITY		✓
	OCCUPANT VISIBILITY	✓	✓
	CRASH- WORTHINESS	✓	

Figure 4.2

GM COMFORT/GT PLEASE FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		PLEASE		
		PROVIDE PHYSICAL COMFORT	PROVIDE MENTAL COMFORT	ENTERTAIN
COMFORT	CONTROLLED CLIMATE	✓	✓	✓
	BODY SUPPORT	✓	✓	✓
	ENHANCED VISION	✓	✓	✓
	CONTROLLED NOISE	✓	✓	✓
	ENHANCED PRIMARY CONTROL	✓	✓	✓
	CONTROLLED ODOR	✓	✓	
	RIDE	✓	✓	✓
	CONVENIENCE	✓	✓	✓

Figure 4.3

		PLEASE		
		PROVIDE PHYSICAL COMFORT	PROVIDE MENTAL COMFORT	ENTERTAIN
ENJOYMENT	AESTHETICS		✓	✓
	EXCITEMENT		✓	✓
	ENTERTAINMENT		✓	✓

GM ENJOYMENT/GT PLEASE FUNCTIONAL DECOMPOSITION CORRESPONDANCE

Figure 4.4

GM ENERGY AND INFO. MGMT./GT FUNCTION FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		VEHICLE FUNCTION									
		TRANSPORT					PROTECT		PLEASE		
		DEVELOP MOTION	CONTROL MOTION	ACCESS VEHICLE	PROVIDE SPACE AND SUPPORT	MAINTAIN VEHICLE'S ABILITY	PROVIDE SAFETY	PROVIDE SECURITY	PROVIDE PHYSICAL COMFORT	PROVIDE MENTAL COMFORT	ENTERTAIN
ENERGY MANAGEMENT	STORE				✓					✓	
	CONVERT	✓								✓	✓
	ALLOCATE/ DISTRIBUTE		✓				✓		✓	✓	✓
	TRANSMIT	✓					✓			✓	✓
	DISSIPATE	✓					✓		✓	✓	✓
INFORMATION MANAGEMENT	ACTIVE INFORMATION PRESENTATION	✓	✓	✓		✓	✓	✓		✓	✓
	STATIC INFORMATION PRESENTATION	✓	✓	✓		✓	✓	✓		✓	✓

Figure 4.5

Management subfunctions found by GM with the Transport, Protect and Please subfunctions.

The third level maps are in Figures 4.6 through 4.10, which occur in the same order as the previous mappings; that is, Figure 4.6 corresponds to the same GM functions as Figure 4.1 of the second level map. A check mark means "corresponds to" or "is a part of".

Due to the generic nature of the GM functions Energy Management and Information Management, their subfunctions are incorporated in many other subfunctions in the Georgia Tech decomposition.

The next mapping involves the functions with the Baseline Parameters. The Baseline Parameters are a set of nine characteristics by which one can distinguish between particular instances of a vehicle. These parameters could be viewed as the variables a designer would vary to specify different vehicle designs. The relationships were chosen based on the strength of effect a change in a variable in a specified direction would have on a particular function. Figures 4.11, 4.12 and 4.13 show the relationships for the second and third level functions, respectively. The arrows beneath the baseline parameters show the specified direction of change to find the effect on a function. Drive Type does not have a specified change direction since it consists only of front wheel drive, rear wheel drive, and four wheel drive. In addition, the QFD mapping only indicates the strength of relationships, not the exact effect. The typical relationships in a QFD involve only strong, moderate and weak relationships. At the second level, certain relationships are listed as a "washout", which means no net effect, though there is a combination of effects. These "washouts" do not show at the third level relationships due to the increased level of detail for the functions. Also, this relationship, as in the previous section, the second level map was used as guide for the third level map and not a restriction. The justification for each relationship in Figures 4.12 and 4.13 are in Appendix C.

The remaining mapping pertains to the interrelationships of functions. This mapping is modeled after the QFD roof; the relationships are specified in terms of strong or weak positive and negative relationships. The relationships were completed for the complete set of second level functions and the third level subset discussed in the application section of this report. The second level function interrelationships are shown in Figure 4.14. Another unusual correspondence type revealed itself in this picture; this the concept of directional relationships. A relationship could be strong one way and weak in the opposite. For instance, consider the relationship between Provide Space and Support and Access Vehicle. How one accesses the vehicle greatly influences how the vehicle space is provided. However, how the space is provided slightly affects how access to the vehicle is rated. This second level map provides clues for the third level relationships needed for the conceptual level vehicle to develop optimization relationships between functions.

GM TRANSPORTATION/GT TRANSPORT FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		TRANSPORT															
		PROVIDE ENERGY		CONTROL MOTION						PROVIDE SPACE AND SUPPORT			ACCESS VEHICLE			MAINTAIN VEHICLE'S ABILITY	
		GENERATE USEFUL ENERGY	DISTRIBUTE ENERGY	MONITOR STATE OF MOTION	CONTROL ENERGY CONVERSION	CHOOSE DIRECTION	STEER VEHICLE	ACCELERATE VEHICLE	DECELERATE VEHICLE	PROVIDE PASSENGER SPACE	PROVIDE CARGO SPACE	PROVIDE ENGINE SPACE	ACCESS PASSENGER SPACE	ACCESS CARGO SPACE	ACCESS ENGINE SPACE	REPAIR MALFUNCTIONS	PERFORM ROUTINE MAINTENANCE
TRANSPORTATION	START CONTROL				✓												
	SHUT OFF CONTROL				✓												
	DIRECTIONAL SELECTION CONTROL					✓											
	ACCELERATION CONTROL							✓									
	BRAKING CONTROL								✓								
	STEERING AND HANDLING CONTROL						✓										

Figure 4.6

GM SAFETY/GT PROTECT FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		PROTECT			
		PROVIDE SAFETY		PROVIDE SECURITY	
		PREVENT MISHAPS	MITIGATE MISHAPS	PREVENT UNDESIRABLE INTRUSION	PREVENT VEHICLE THEFT
SAFETY	ACCIDENT AVOIDANCE	✓			
	HAZARD PROTECTION	✓	✓		
	SECURITY			✓	✓
	OCCUPANT VISIBILITY	✓			
	CRASH- WORTHINESS		✓		

Figure 4.7

GM COMFORT/GT PLEASE FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		PLEASE											
		PROVIDE PHYSICAL COMFORT							PROVIDE MENTAL COMFORT			ENTERTAIN	
		MAINTAIN AMBIENT TEMPERATURE	PROVIDE COMFORTABLE SPACING	PROVIDE ERGONOMIC INTERFACES	PROVIDE COMFORTABLE VIEW	PROVIDE COMFORTABLE AIR	PROVIDE COMFORTABLE RIDE	PROVIDE LOW NOISE	PERFORM AS EXPECTED	PROVIDE FEELING OF VALUE	PROVIDE APPROPRIATE LOOK	ALLOW FOR EXHILARATION	ALLOW FOR STIMULATING ENVIRONMENT
COMFORT	CONTROLLED CLIMATE	✓							✓	✓			✓
	BODY SUPPORT			✓					✓	✓			✓
	ENHANCED VISION				✓				✓	✓			✓
	CONTROLLED NOISE							✓	✓	✓			✓
	ENHANCED PRIMARY CONTROL			✓					✓	✓		✓	
	CONTROLLED ODOR					✓			✓	✓			
	RIDE						✓		✓	✓		✓	
	CONVENIENCE			✓					✓	✓			✓

Figure 4.8

GM ENJOYMENT/GT PLEASE FUNCTIONAL DECOMPOSITION CORRESPONDANCE

		PLEASE											
		PROVIDE PHYSICAL COMFORT							PROVIDE MENTAL COMFORT			ENTERTAIN	
		MAINTAIN AMBIENT TEMPERATURE	PROVIDE COMFORTABLE SPACING	PROVIDE ERGONOMIC INTERFACES	PROVIDE COMFORTABLE VIEW	PROVIDE COMFORTABLE AIR	PROVIDE COMFORTABLE RIDE	PROVIDE LOW NOISE	PERFORM AS EXPECTED	PROVIDE FEELING OF VALUE	PROVIDE APPROPRIATE LOOK	ALLOW FOR EXHILARATION	ALLOW FOR STIMULATING ENVIRONMENT
ENJOYMENT	AESTHETICS									✓	✓		✓
	EXCITEMENT								✓	✓		✓	✓
	ENTERTAINMENT								✓	✓			✓

Figure 4.9

**GM ENERGY AND INFO. MGMT./GT FUNCTION
FUNCTIONAL DECOMPOSITION CORRESPONDANCE**

		TRANSPORT													PROTECT				PLEASE														
		PROVIDE ENERGY		CONTROL MOTION					PROVIDE SPACE AND SUPPORT			ACCESS VEHICLE		MAINTAIN VEHICLE'S ABILITY		PROVIDE SAFETY		PROVIDE SECURITY		PROVIDE PHYSICAL COMFORT						PROVIDE MENTAL COMFORT		ENTERTAIN					
		GENERATE USEFUL ENERGY	DISTRIBUTE ENERGY	MONITOR STATE OF MOTION	CONTROL ENERGY CONVERSION	CHOOSE DIRECTION	STEER VEHICLE	ACCELERATE VEHICLE	DECELERATE VEHICLE	PROVIDE PASSENGER SPACE	PROVIDE CARGO SPACE	PROVIDE ENGINE SPACE	ACCESS PASSENGER SPACE	ACCESS CARGO SPACE	ACCESS ENGINE SPACE	REPAIR MALFUNCTIONS	PERFORM ROUTINE MAINTENANCE	PREVENT MISHAPS	MITIGATE MISHAPS	PREVENT UNDESIRABLE INTERUSION	PREVENT VEHICLE THEFT	MAINTAIN AMBIENT TEMPERATURE	PROVIDE COMFORTABLE SPACING	PROVIDE SMOOTHING INTERFACES	PROVIDE COMFORTABLE VIEW	PROVIDE COMFORTABLE AIR	PROVIDE COMFORTABLE NOISE	PROVIDE LOW NOISE	PERFORM AS EXPECTED	PROVIDE FEELING OF VALUE	PROVIDE APPROPRIATE LOCK	ALLOW FOR EXPLANATION	ALLOW FOR STIMULATING ENVIRONMENT
ENERGY MANAGEMENT	STORE									✓																		✓	✓			✓	
	CONVERT	✓						✓																				✓	✓		✓		
	ALLOCATE/DISTRIBUTE		✓		✓		✓	✓				✓	✓				✓		✓	✓	✓		✓			✓	✓	✓	✓	✓		✓	✓
	TRANSMIT		✓		✓		✓	✓				✓	✓				✓		✓	✓	✓	✓		✓		✓	✓	✓	✓		✓	✓	
	DISSIPATE		✓		✓		✓	✓				✓	✓				✓		✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		✓	✓
INFORMATION MANAGEMENT	ACTIVE INFORMATION PRESENTATION			✓								✓	✓				✓		✓	✓	✓					✓		✓	✓	✓	✓	✓	✓
	STATIC INFORMATION PRESENTATION			✓								✓	✓	✓	✓	✓	✓		✓	✓	✓	✓				✓		✓	✓	✓	✓	✓	✓

Figure 4.10

FUNCTION AND BASELINE PARAMETERS RELATIONSHIPS

- ◎ STRONG RELATIONSHIP
 ○ MODERATE RELATIONSHIP
 △ WEAK RELATIONSHIP
 ~ WASH OUT

		NUMBER OF PEOPLE ↑	NUMBER OF DOORS ↑	ENGINE SIZE ↑	CARGO CAPABILITY ↑	WHEEL BASE ↑	DRIVE TYPE	MATERIAL PERCENTAGE ↑ % Metal	SILHOUETTE ↓ C&	CLEARANCE ↑
TRANSPORT	DEVELOP MOTION	◎		◎	◎		◎	◎	○	△
	CONTROL MOTION	○		○	△	○	◎	△	△	△
	PROVIDE SPACE AND SUPPORT	◎	△	◎	◎	◎	△	△	○	○
	ACCESS VEHICLE	◎	◎		△	△			○	△
	MAINTAIN VEHICLE'S ABILITY		~	◎	△		◎	○	○	◎
PROTECT	PROVIDE SAFETY	◎	△	◎	○	○	○			○
	PROVIDE SECURITY		◎		~					
PLEASE	PROVIDE PHYSICAL COMFORT	◎	△		○	△	~	~		~
	PROVIDE MENTAL COMFORT	△	~	◎	△	△	△	△	◎	~
	ENTERTAIN	△		◎		~	○		◎	~

Figure 4.11

- ◎ STRONG RELATIONSHIP
 ○ MODERATE RELATIONSHIP
 △ WEAK RELATIONSHIP

			NUMBER OF PEOPLE ↑	NUMBER OF DOORS ↑	ENGINE SIZE ↑	CARGO CAPABILITY ↑	WHEEL BASE ↑	DRIVE TYPE	MATERIAL PERCENTAGE ↑ % Metal	SILHOUETTE ↕	CLEARANCE ↑
TRANSPORT	PROVIDE ENERGY	GENERATE USEFUL ENERGY	◎	△	◎	◎		○	◎	◎	△
		DISTRIBUTE ENERGY	◎	△	◎	◎	○	◎	◎	○	△
	CONTROL MOTION	MONITOR STATE OF MOTION	○	△		△	△	△		△	△
		CONTROL ENERGY CONVERSION			◎						
		CHOOSE DIRECTION						○			
		STEER VEHICLE	△	△	△	△	◎	○	△		△
		ACCELERATE VEHICLE	◎	△	◎	◎	△	△	◎	○	△
		DECELERATE VEHICLE	◎		△	◎	△	○	◎		
	PROVIDE SPACE AND SUPPORT	PROVIDE PASSENGER SPACE	◎		○	△	◎	△	△	△	△
		PROVIDE CARGO SPACE	○		△	◎	△		△	○	△
		PROVIDE ENGINE SPACE	○		◎	◎	△	◎	○	◎	△
	ACCESS VEHICLE	ACCESS PASSENGER SPACE	◎	◎		△	△	△		△	△
		ACCESS CARGO SPACE	△	○		◎				○	△
		ACCESS ENGINE SPACE			◎			○		△	○
	MAINTAIN VEHICLE'S ABILITY	REPAIR MALFUNCTIONS		△	◎			◎		○	◎
		PERFORM ROUTINE MAINTENANCE		△	◎	△		◎	△	○	◎

Figure 4.12 Function and Baseline Parameters for Transport, Third Level

			◎ STRONG RELATIONSHIP ○ MODERATE RELATIONSHIP △ WEAK RELATIONSHIP								
			NUMBER OF PEOPLE ↑	NUMBER OF DOORS ↑	ENGINE SIZE ↑	CARGO CAPABILITY ↑	WHEEL BASE ↑	DRIVE TYPE	MATERIAL PERCENTAGE ↑ % Metal	SILHOUETTE ↓	CLEARANCE ↑
PROTECT	PROVIDE SAFETY	PREVENT MISHAPS	◎		◎	△	○	○	△	○	○
		MITIGATE MISHAPS	◎	△	○	○	○	○	○	△	
	PROVIDE SECURITY	PREVENT UNDESIRABLE INTRUSION		◎		○				△	
		PREVENT VEHICLE THEFT		○						△	
	PROVIDE PHYSICAL COMFORT	MAINTAIN AMBIENT TEMPERATURE	◎		△		△			◎	
		PROVIDE COMFORTABLE SPACING	◎	○		○	○	△	△	△	△
		PROVIDE ERGONOMIC INTERFACES	◎	◎		○		○		△	△
		PROVIDE COMFORTABLE VIEW	○	△		○	△			◎	△
		PROVIDE COMFORTABLE AIR	◎							○	
		PROVIDE COMFORTABLE RIDE	○		◎	○	○	△	○		○
		PROVIDE LOW NOISE			◎			○			○
	PROVIDE MENTAL COMFORT	PERFORM AS EXPECTED	○	○	○	○	○	○		○	○
		PROVIDE FEELING OF VALUE	◎		◎	◎	△		○	○	
		PROVIDE APPROPRIATE LOOK	○	○	○	△	○	○	○	◎	○
	ENTERTAIN	ALLOW FOR EXHILARATION	○		◎		◎	◎		○	○
		ALLOW FOR STIMULATING ENVIRONMENT			○				△	◎	

Figure 4.13 Function and Baseline Parameters for Please, Protect,
Third Level

FUNCTION INTERRELATIONSHIPS

- ◎ STRONG POSITIVE
 ○ WEAK POSITIVE
 × WEAK NEGATIVE
 * STRONG NEGATIVE

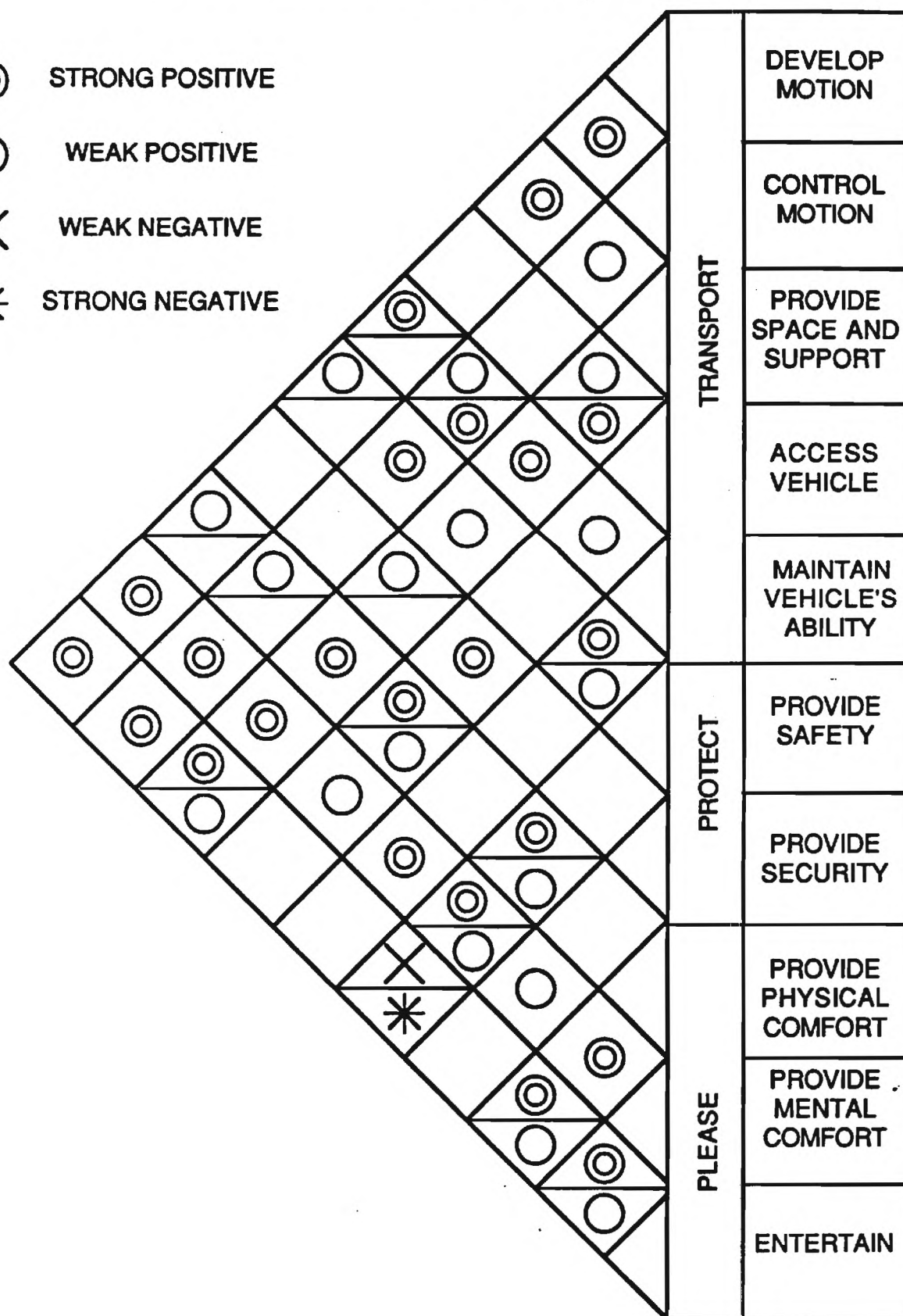


Figure 4.14

4.3 Conceptual Level Application

This section reports the areas that have been completed in the area of applying the conceptual level to the design of the vehicle.

4.3.1 Functional Description Subset

The third level functions were mapped to the baseline parameters to determine the minimum set of functions necessary to describe a conceptual level vehicle. This will allow us to test the methodology without being burdened with the whole set. This first cut produced nine functions. It was observed that none of the nine strongly correlated to the parameter "Number of Doors". Therefore, the function "Access Passenger Space" was added to the list. At this time, the overall input and output was determined and is discussed in Chapter 5. On the basis of the overall output and group discussions, seven more functions were added to the list. The following is a listing of the functions:

<u>FUNCTION</u>	<u>TYPE</u>
F1 Transport	
F1.1 Provide Energy	
F1.1.1 Generate Useful Energy	Enable
F1.1.2 Distribute Energy	Enable
F1.2 Control Motion	
F1.2.4 Steer Vehicle	Direct
F1.2.5 Accelerate Vehicle	Direct
F1.2.6 Decelerate Vehicle	Direct
F1.3 Provide Space and Support	
F1.3.1 Provide Passenger Space	Enable
F1.3.2 Provide Cargo Space	Enable
F1.4 Access Vehicle	
F1.4.1 Access Passenger Space	Direct
F1.4.2 Access Cargo Space	Direct
F2 Protect	
F2.1 Provide Safety	
F2.1.1 Prevent Mishaps	Both
F2.1.2 Mitigate Mishaps	Both

F3 Please

F3.1 Provide Physical Comfort	
F3.1.2 Provide Comfortable Spacing	Enable
F3.2 Provide Mental Comfort	
F3.2.1 Perform As Expected	Enable
F3.2.2 Provide Feeling of Value	Enable
F3.2.3 Provide Appropriate Look	Enable
F3.3 Entertain	
F3.3.1 Allow for Exhilaration	Both

The concept of direct/enable functions indicates a preliminary method for translating the customer/designer data into the functional attributes. The direct function attributes should be determined first and then the enabling function attributes would be determined and optimized. This would be based on the functional interrelationships and their associated weights for the objective function. This would handle the variety of Entertain profiles, which vary greatly from customer to customer. Certain functions have both direct and enable subfunctions. These are reflected in the choice of functional attributes for these functions which are pertinent to a conceptual level vehicle. The following is a list of the function subset with the initial functional attributes chosen:

<u>FUNCTION</u>	<u>ATTRIBUTE(S)</u>
F1 Transport	
F1.1 Provide Energy	
F1.1.1 Generate Useful Energy	Horsepower
F1.1.2 Distribute Energy	Type
F1.2 Control Motion	
F1.2.4 Steer Vehicle	Turn Radius
F1.2.5 Accelerate Vehicle	0-60 Time
F1.2.6 Decelerate Vehicle	60-0 Distance
F1.3 Provide Space and Support	
F1.3.1 Provide Passenger Space	Volume (L,W,H)
F1.3.2 Provide Cargo Space	Volume (L,W,H)
F1.3.3 Provide Engine Space	Volume (L,W,H)
F1.4 Access Vehicle	
F1.4.1 Access Passenger Space	# Doors, Area
F1.4.2 Access Cargo Space	Area
F2 Protect	
F2.1 Provide Safety	
F2.1.1 Prevent Mishaps	Std.-Great
Based on F1.2.4, F1.2.5, F1.2.6, and Clearance	
F2.1.2 Mitigate Mishaps	Min.-Best
Based on F1.2.4, F1.2.6, F1.3.2, and F1.3.3	

F3 Please**F3.1 Provide Physical Comfort**

F3.1.2 Provide Comfortable Spacing	Volume, Human Percentile
---	-----------------------------

F3.2 Provide Mental Comfort

F3.2.1 Perform As Expected	Evaluate @ end
-----------------------------------	----------------

F3.2.2 Provide Feeling of Value	Evaluate @ end
--	----------------

F3.2.3 Provide Appropriate Look	Silhouette, Layout
--	-----------------------

F3.3 Entertain

F3.3.1 Allow for Exhilaration	Std.-Great
--------------------------------------	------------

Based on F1.2.4, F1.2.5, and F1.2.6.

4.3.2 Function Relationships

The function subset interrelationships has been specified in the form of a QFD roof and is presented in Figure 4.15. Note that there are no directional relationships. Further examination will assist in the development of the function optimization scheme. This will require the use of a customer scenario. The scenario for this examination will concern the previous purchase of a 1987 Chevrolet Nova. The customer's previous ideas have been placed in the pertinent input format. See Figure 4.16.

4.4 Summary

This chapter has defined the vehicle in terms of its functions to three levels of detail. This provides the necessary information for the preliminary assessment of the functional description of the automobile. The selection of a subset will allow us to test our methodology by using the function to describe a vehicle and to follow the thread from QFD to Form. It will also allow us to set up the function vector necessary to optimize the functional description of the vehicle. This vector will be in form of

$$\Phi = \alpha \text{ Transport} + \beta \text{ Please} + \gamma \text{ Protect.}$$

The next step is to map the customer's or designer's data to the associated functions and determining the appropriate relationships. A baseline car will first will be defined. This will be done by looking for the customer data that supports each functional attribute. With this customer scenario, the functional attributes will be determined for an example function to develop the relationships tying the Voice of the Customer, Baseline Parameters, and functions. Also, specifying these relationships will lead to developing a feed forward function attribute optimization. During form development, there may be no suitable combination of forms to fulfill the functions. This will require further negotiation and optimization of the functional attributes based on the customer's priorities. This process will allow the understanding of the front end process through to functional specification.

VOICE OF THE CUSTOMER INPUT
1987 Chevrolet Nova

<u>ITEM</u>	<u>VALUE/DESCRIPTION</u>
Body Type	(Don't Care)
Number of Side Doors	(Don't Care)
Overall Vehicle Size	Small
Seating Capacity	4
Passenger Volume	Minimum +
Cargo Volume	Normal
Cost	Economy + (10k)
Silhouette	Round/Streamlined
Acceleration	Minimum +
Fuel Economy	Good
Handling/Ride	Middle(Comfort/Response)

Figure 4.16

Chapter 5 PHYSICAL PERSPECTIVE OF THE VEHICLE

5.1 Introduction

Vehicle form description plays the other important role in the OOVm. The form description of a design specifies how the design will do what is required, i.e., how it meets the functional requirements. Form description does not (and should not) describe what the design is to do, but rather how the design is to be accomplished. This is an important point. We want to ensure that the conceptual vehicle form characteristics follow from the functional requirements (and, ultimately, from the customer requirements) and not the other way around. That is, we want to design the vehicle to fulfill the customer requirements (to the fullest extent possible), not specify the functional requirements by the forms chosen. This point may seem obvious, but automobile manufacturers have been accused for years of designing vehicles with little or no regard for what the customer wants, then relying on marketing strategies to "adapt" the customer to the car (i.e., convince the customer this is what he or she wants). In today's fiercely competitive automotive market, however, such strategies are finding little success. We need to understand what the customer wants, then "adapt" the car to the customer.

Vehicle form description plays another important role in the OOVm. It was discussed above that we need to determine the decisions that must be made to design the conceptual vehicle. We must also determine, however, the information needed to make these decisions. By examining the relationships of the vehicle forms to the functional requirements (and to other forms), we can make this determination.

This chapter discusses the accomplishments made in vehicle form description as well as work needed to complete the OOVm.

5.2 Vehicle Form Description

5.2.1 Input Parameters

Working from the set of baseline parameters discussed in Section 3.2, we developed the set of input parameters to the OOVm. The input parameters are the set of variables for which a customer specifies values to describe the vehicle he or she wants. That is, the inputs are a description of all the parameters that compose the "voice of the customer" (i.e., customer attributes), at least at the conceptual vehicle level. The values specified for the input parameters will serve as the requirements according to which (along with constraints such as government regulations and technological limitations), for example, the vehicle will be designed. We designed as much flexibility into the inputs as possible, to enable a person to describe a convertible sports-car as easily as an economy car or off-road vehicle. We

drew up the list of inputs by debating amongst ourselves and conducting interviews (asking questions such as "Why did you buy that car?" and "What are you going to look for in you next car?"). We also talked to people in the automotive industry. For example, we went to local dealers that sell many different makes and types of vehicles (Cadillac, Pontiac, Jaguar, Land Rover) and got their input on what people want in their cars (and what sells cars). All this "input" went into our input list.

The input parameter list is shown below. An asterisk (*) beside an input parameter signifies that it is also a baseline parameter. In parentheses next to each parameter are candidate qualitative/quantitative values for the customer to input. For example, though, for overall vehicle size, each qualitative size choice (small, compact, medium, large) will be displayed with a corresponding quantitative range of wheelbase length. In addition to asking for values to be specified for the input parameters, the OOVm will also prompt the customer to rank the importance of all the parameters to him or her. In this way, if conflicts between requirements arise in the design process, we'll have a "voice" of relative importance rankings telling us where to make trade-offs so that we may design the optimal vehicle. This will serve as the input to the optimization vectors.

INPUT PARAMETERS

Body Type (sedan, coupe, hatchback, station wagon, minivan, utility, sport, convertible)
Number of Side Doors* [2, 3 (2 + 1 sliding), 4]
Overall Vehicle Size (small, compact, medium, large / ranges of wheelbase)
Seating Capacity* (2, 3, 4, 5, 6, 6+)
Passenger Volume (minimum necessary to spacious / ranges of volume in cubic feet)
Cargo Volume* (none to maximum feasible / ranges of volume in cubic feet)
Cost (economy to luxury / ranges of price)
Silhouette* (straight to round and box to streamlined)
Acceleration (minimum necessary to maximum feasible / ranges of 0 to 60 mph times)
Fuel Economy (ranges of combined city and highway mpg)
Handling / Ride (maximum comfort to maximum response)
Relative Importance Rankings of Input Variables (along with safety and reliability)

* Baseline Parameter

Additionally, the customer should have the option to specify desired values (or types) for any of the output variables.

5.2.2 Output Parameters

Again working from the set of baseline parameters, we developed the set of output parameters from the OOVm. The output parameters are the set of variables (i.e., product characteristics) whose values define a vehicle in technical terms, again at the conceptual level. We formatted the output list on the basis of what the automotive industry uses to define a car technically. The list reflects those specifications most commonly given by car manufacturers (e.g., in sales brochures), consumer organizations (e.g., Consumer Reports), and enthusiast magazines (e.g., Car and Driver, Road & Track, Motor Trend, etc.) in their descriptions of vehicles. It is the goal of the OOVm to assign values to these "design variables" that will fulfill the requirements specified by the customer.

The output parameter list is shown below. Additionally, we want to have a pictorial output that shows the silhouette of the vehicle and, possibly, the relative sizes and locations of the subsystems beneath. A single asterisk (*) besides an output parameter signifies that it is also a baseline parameter. The baseline parameter "Material Percentage (% Metal)" is not defined explicitly on either the input or output parameter list. The output parameters that have a double asterisk (**), however, are candidate parameters from whose values material percentage may be calculated. We have chosen a subset of the output parameters to be used to determine if our OOVm methodology is correct. A triple asterisk (***) is shown by each of the output parameters that compose this subset. After testing the OOVm with the subset, the entire set will be implemented in the model.

OUTPUT PARAMETERS

VEHICLE TYPE:

Body Type*** (sedan, coupe, hatchback, station wagon,
minivan, utility, sport, convertible)

Number of Side Doors*,*** (2, 3, 4)

Seating Capacity*,*** (2, 3, 4, 5, 6, 6+)

Engine Location and Orientation (front, mid, rear and
longitudinal, transverse)

Driven Wheels*,*** (front, rear, all)

PRICE:

Estimated Cost***

GENERAL DATA:

Wheelbase*

Track (front and rear)

Length***

Width***

Height

Ground Clearance*,***

Weight,*****

Weight Distribution (front and rear)

Fuel Tank Capacity

INTERIOR:

Passenger Volume* (and height, length, width of volume)**

Cargo Volume*,* (and height, length, width)**

ENGINE:

Type -- then if internal-combustion, reciprocating-piston, and four-stroke:

Subtype (spark-ignition, diesel) -- then if spark-ignition:

Aspiration Type (normal, supercharged, turbocharged)

Engine Block Material**

Cylinder Head Material**

Number of Cylinders

Cylinder Arrangement (in-line, vee, opposed)

Number of Valves

Number of Camshafts

Camshaft Location [engine block (push-rod), cylinder head (overhead)]

Displacement*

Bore and Stroke

Compression Ratio

Fuel Management Type (carburetor, single-point fuel injection, multi-point fuel injection)

Cooling Type (air, water)

Horsepower***

Torque

DRIVETRAIN:

Transmission Type (manual, automatic)

Number of Forward Speeds

Final Drive Ratio

CHASSIS AND BODY:

Structure Type and Material**

Body Panel Material**

Suspension Type (front and rear)

Brake Type and Size (front and rear)

Wheel and Tire Type and Size (front and rear)

Steering Type

Turning Circle Diameter

PERFORMANCE DATA:

Acceleration (0 to 30 mph, 0 to 60 mph, 45 to 65 mph passing, 1/4 mile, and top speed)

Fuel Economy (city and highway)
Braking (60 to 0 mph)
Handling (roadholding on 300 ft diameter skidpad)

- * Baseline Parameter
- ** Candidates to combine to determine the Baseline Parameter "Material Percentage (% Metal)"
- *** Methodology test subset

A few points about the input and output parameter lists should be made. First, both lists contain a mixture of function and form parameters. For example, the input list contains the form parameter "number of side doors" and the function parameter "acceleration time" and the output list includes the form parameter "engine type" and the function parameter "braking distance." Our research is based on the premise that form follows function, so the fact that the input to and output from the OOVm is a mixture of the two is seemingly inconsistent with our proposition. However, the rationale for specifying the lists as mixtures of function and form is to reflect the reality that vehicle purchasers and manufacturers do, indeed, describe cars using a mixture of both types of parameters. Our assertions are that forms specified by customers are for functional purposes and functions specified in technical descriptions of vehicles are the result of chosen form parameters. For example, the input parameter "number of side doors" can be translated into the functional requirement "access vehicle" and the output functional parameter "braking distance" is a result of the "brake type" form chosen (and other factors). The OOVm will implement the above proposition and assertions by translating all customer requirements into functional requirements, then translate these functional requirements into forms and, finally, derive the functional characteristics of the vehicle from the forms (and combination of forms) chosen.

Another point to note is that the input and output lists contain some (seemingly) repetitive data. For example, both lists contain the parameter "acceleration." Again, this is in part due to the nature of how customers and manufacturers describe their vehicles -- that is, sometimes they use the same parameters. The repetition of parameters in the input and output lists isn't necessarily redundant. The inputs (i.e., customer requirements) are generally qualitative, whereas the outputs are necessarily quantitative. For example, the customer may specify as input that he or she wants "fast" acceleration. The OOVm will translate this into a quantitative function to be fulfilled, the appropriate forms will be chosen to fulfill this function (and all other functional requirements) and the quantitative acceleration characteristics of the resultant vehicle will be output. Even where an input to the system is qualitative, the same output parameter may not have the same qualitative value where optimization was necessary due to conflicting requirements or constraints.

The final point is that the car specified by values of the output parameters should be the "same" as the car specified by values of the input parameters. The vehicle design process is simply a conversion of what the customer says he or she wants into a vehicle that fulfills those requirements. If the design is successful, the input to and output from our system will just be different perspectives (i.e., views) of the same thing -- the first describing the "desired" vehicle from the standpoint of the customer, the second describing the same but "realized" vehicle in technical terms.

5.3 Vehicle Form Decomposition

As stated above, the goal of the OOVm is to assign values to the output "design variables" such that the car designed optimally fulfills the requirements specified by the customer and meets all constraints. Before we can implement the OOVm, we need to structure the organization of the design process, i.e., the decision making process through which customer requirements are translated into product characteristics. We need to identify all the steps the system will have to go through in order to design the vehicle. Furthermore, we need to find the optimal sequencing of those steps so as to make the design feedforward and minimize the iterations necessary to converge upon an acceptable solution. This is important because the vehicle (even at the conceptual level) is such a complex system that its design can easily become intractable.

One way of making these determinations is by examining the relationships among the forms that compose a vehicle. Hierarchical decomposition (further details of which are discussed in latter chapters of this report), given the qualitative relationships among subsystems of a complex system, will use a sorting algorithm to sequence the order in which the subsystems should be designed so as to localize interdependencies and minimize the amount of feedback and iteration required in order to converge upon an optimal solution. It has been proposed that this method be utilized to establish the order in which the forms should be designed, aggregated, and optimized into an overall vehicle. Before we can put this method to use, though, we must define what the forms are that compose a vehicle and then what the qualitative relationships are amongst these forms.

5.3.1 Form Decomposition

The vehicle was decomposed into a form hierarchy. The hierarchical form decomposition was developed using a systems approach, that is, the vehicle was decomposed into groups (or "systems") of forms according to the common functions the forms perform. Functional groupings of forms will allow a more direct mapping from function to forms and localize the forms within the OOVm that may have to be optimized in order to fulfill a desired function. The decomposition also gives us a good idea of how the

forms, which are designed to fulfill the functional requirements, will be aggregated into an overall vehicle. The top-level form is, of course, the overall vehicle "system." The first-level decomposition of the overall vehicle system yielded four sub-systems :

1. Transportation/Support Systems
(make the vehicle "mobile")
2. Power Generation/Transformation Systems
(make the vehicle "auto"-mobile)
3. Entertainment/Driver Interface Systems
(make the vehicle "people" mobile)
4. Computer Control Systems
(link the previous three systems)

Decomposition of the first-level forms yielded eight "sub" sub-systems, or second-level forms:

1. Transportation/Support Systems
 - 1.1 Structural Systems
 - 1.2 Chassis Systems
2. Power Generation/Transformation Systems
 - 2.1 Power Generation (Engine) Systems
 - 2.2 Power Transmission/Transformation Systems
3. Entertainment/Driver Interface Systems
 - 3.1 Interior Systems
 - 3.2 Exterior Systems
4. Computer Control Systems
 - 4.1 Integration Systems
 - 4.2 External Interface Systems

At the conceptual level, differentiation between vehicles occurs by varying the quality and quantity of the design variables associated with these forms. The values assigned to the input parameters (i.e., customer attributes) by the customer will determine the values of each of the functions generated by the functional decomposition. These in turn will determine the values of the form variables. Finally, these values will determine the values of the output parameters.

The General Motors Uniform Parts Classification (UPC) was used to test the completeness of the vehicle form decomposition. That is, we went through the UPC and ensured that each of the parts in the list could be "mapped" or classified into one of the vehicle systems. This check was important -- it insures that the form decomposition for the OOVm at the conceptual level fully describes the vehicle and will be readily expandable for detailed design. The UPC also helped in the development of the decomposition. We looked at each of the parts in the list and asked "What does this part do?" By abstracting then aggregating the functions performed by the parts, we "built-up" the systems that compose the conceptual vehicle.

One question that is often asked when the decomposition is presented for review is "Where is the electrical system?" The electrical system performs many functions; hence, it is many places in our decomposition. For example, the alternator converts mechanical to electrical power; therefore, it is part of the power transmission/transformation system. Likewise, the wiring harness is part of this system because it "transmits" electricity from one part of the vehicle to another. The vehicle "computer" and its associated sensors and wiring (wiring for the purpose of transmitting and receiving information, not power) are part of the computer control system because they "function" as controllers of the other systems. Again, this functional grouping of forms will allow a more direct mapping from function to form. Note that in chapter 8 we discuss alternative views of the OOV. This capability will allow designers concerned with the electrical system to "view" what they are concerned with as a whole entity and ensure its completeness and functionality.

5.3.2 Form Relationships

Similar to the baseline parameters, the interactions among the vehicle forms were explored through the use of the QFD house roof relationship matrix. Figure 5.1 illustrates the interactions among the first-level forms of the vehicle decomposition. Note that there is quite a bit of directionality present among the interactions. For example, the power generation/transformation system has a weak, positive effect on the entertainment/driver interface system (e.g., with more power you can have a bigger radio), but there is no relationship in the opposite direction. The interactions among the second-level forms of the decomposition were also investigated. These are shown in Figure 5.2. It is these qualitative relationships that will be used in hierarchical decomposition to establish the sequence in which the systems should be designed and synthesized into an overall vehicle system.

The relationships between the functions and forms (from the decompositions) were also investigated. Figure 5.3 shows the relationships of the first-level functions to the first-level forms. Here, we used the QFD house "body" relationship matrix with symbols to indicate the strength (strong, moderate, weak, or none) of the relationship. The symbols reflect the effect that modifying a function will have on form. Note that the matrix is filled -- this indicates that there is quite a lot of coupling. The relationships between the second-level functions and second-level forms were also determined. These are shown in Figure 5.4. Note that the relationships between the function "entertain" and the forms were omitted. This is because the relationships are highly dependent on the specific customer involved. For example, one driver may derive all his entertainment from the radio, in which instance there would be a strong relationship between the function entertain and the interior system.

FIRST-LEVEL FORM INTERACTIONS

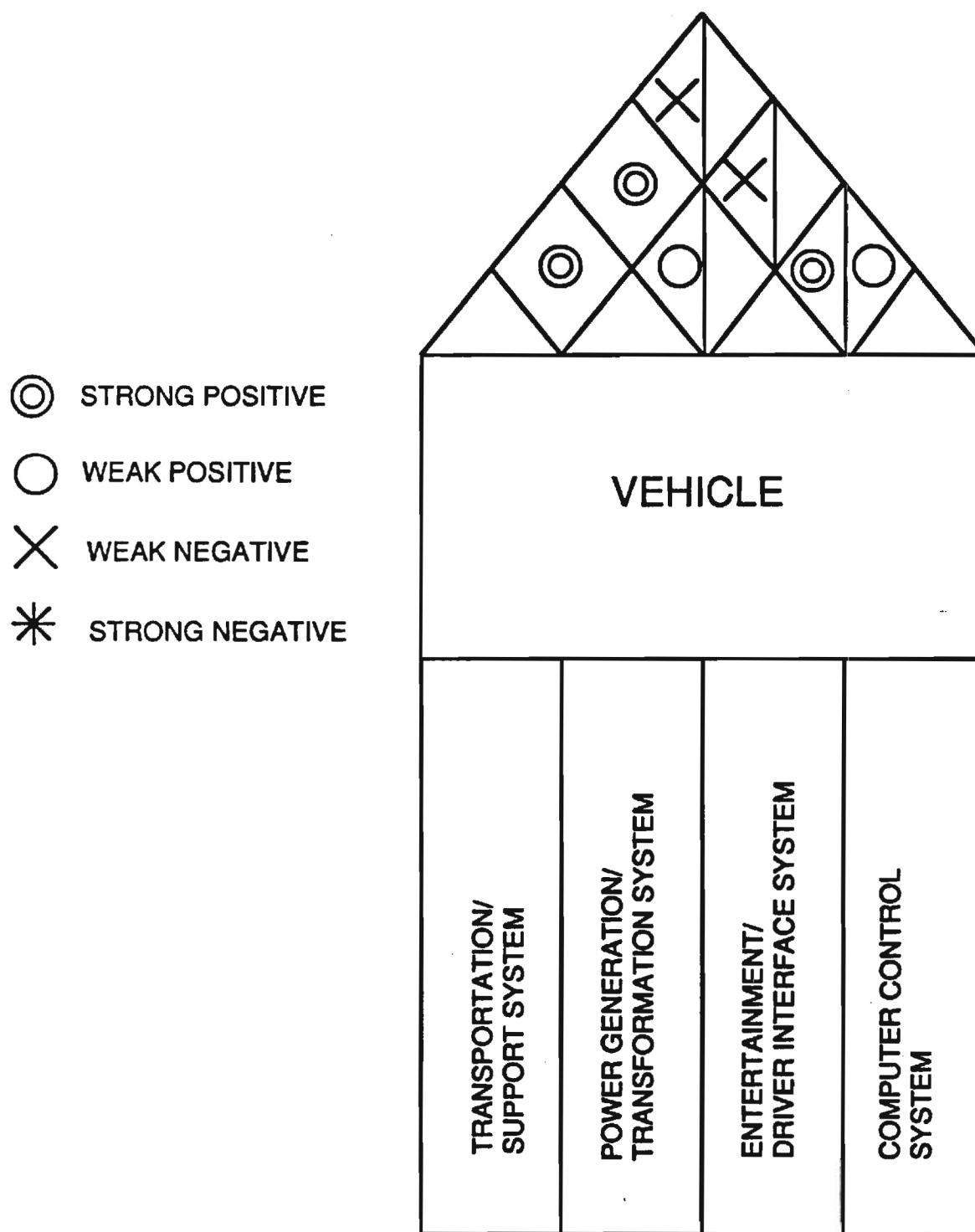


Figure 5.1

SECOND-LEVEL FORM INTERACTIONS

- ⊙ STRONG POSITIVE
 ○ WEAK POSITIVE
 ✕ WEAK NEGATIVE
 ✱ STRONG NEGATIVE

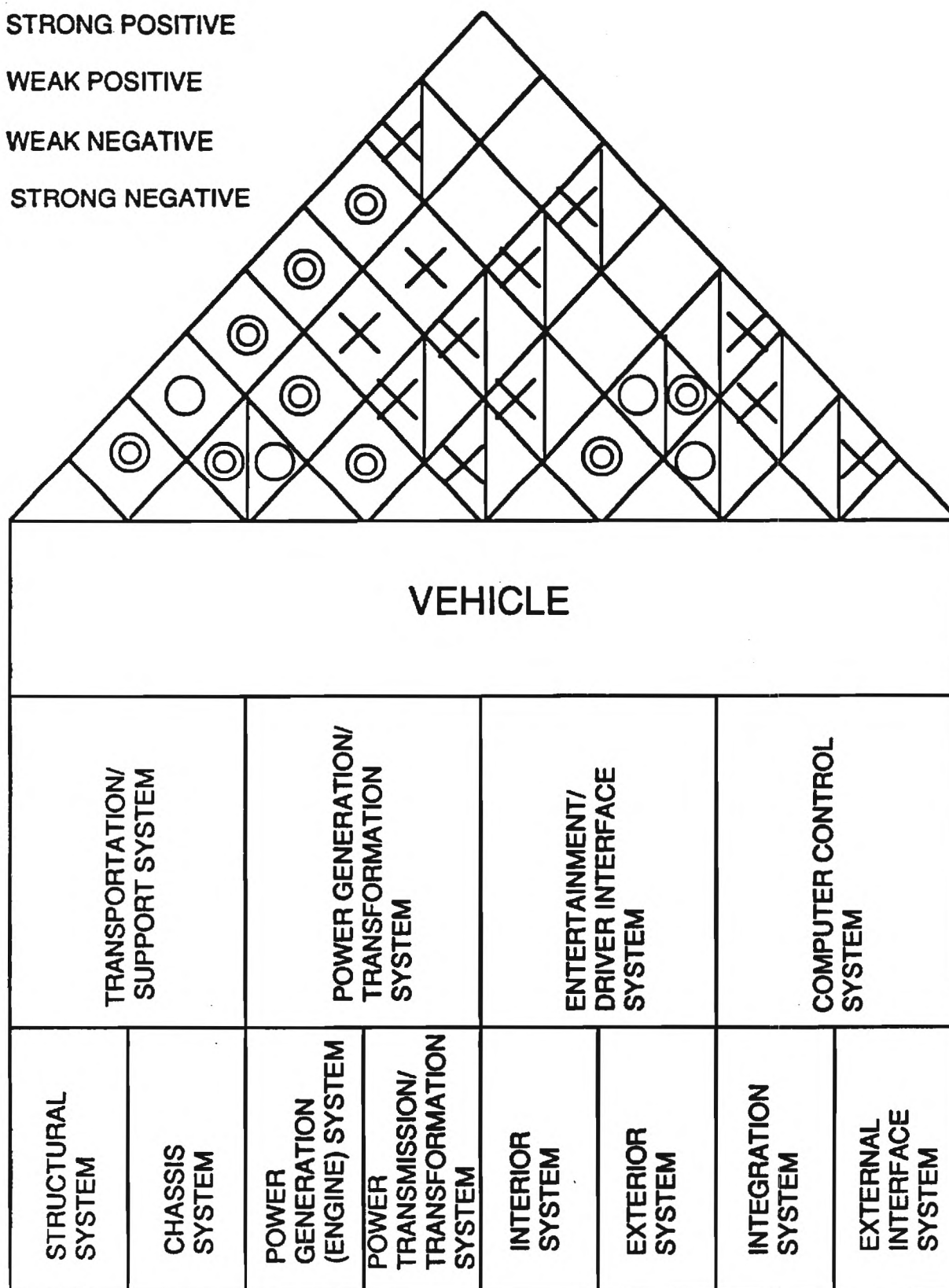





Figure 5.2

FIRST-LEVEL FUNCTION TO FIRST-LEVEL FORM RELATIONSHIPS

-  STRONG RELATIONSHIP
 MODERATE RELATIONSHIP
 WEAK RELATIONSHIP













VEHICLE				
	TRANSPORTATION/ SUPPORT SYSTEM	POWER GENERATION/ TRANSFORMATION SYSTEM	ENTERTAINMENT/ DRIVER INTERFACE SYSTEM	COMPUTER CONTROL SYSTEM
TRANSPORT				
PROTECT				
PLEASE				

Figure 5.3

SECOND-LEVEL FUNCTION TO SECOND-LEVEL FORM RELATIONSHIPS

		TRANSPORTATION/ SUPPORT SYSTEM		POWER GENERATION/ TRANSFORMATION SYSTEM		ENTERTAINMENT/ DRIVER INTERFACE SYSTEM		COMPUTER CONTROL SYSTEM	
		STRUCTURAL SYSTEM	CHASSIS SYSTEM	POWER GENERATION (ENGINE) SYSTEM	POWER TRANSMISSION/ TRANSFORMATION SYSTEM	INTERIOR SYSTEM	EXTERIOR SYSTEM	INTEGRATION SYSTEM	EXTERNAL INTERFACE SYSTEM
TRANSPORT	PROVIDE ENERGY	◎		◎			△	○	
	CONTROL MOTION	△	◎	○	△	◎		◎	
	ACCESS VEHICLE	◎	△			○	◎	△	
	MAINTAIN VEHICLE'S ABILITY		○	◎	△	△	△	○	○
PROTECT	PROVIDE SAFETY	◎	◎	○	○	◎	○	○	
	PROVIDE SECURITY	◎	△			△	◎	△	
PLEASE	PROVIDE PHYSICAL COMFORT	○	○	△	△	◎		◎	
	PROVIDE MENTAL COMFORT	○	△	○	△	◎	◎	△	
	ENTERTAIN								
RELATIONSHIPS RELATIVE TO SPECIFIC CUSTOMER									

Figure 5.4




Another driver may be entertained by a car that can take corners fast. In this case, there would be a strong relationship between entertain and the chassis system.

The relationships between the baseline parameters and forms were also studied. These relationships show the effects the vehicle forms have on the nine basic characteristics that describe the conceptual vehicle. The relationships of the baseline parameters to the first-level forms are shown in Figure 5.5, and to the second-level forms in Figure 5.6. Note that the matrices are almost entirely filled, except that there are no relationships between the baseline parameters and the computer control system (and its sub-systems). This is because the control system is basically an interface between the other systems -- in and of itself, it doesn't affect the external characteristics of the vehicle.

To implement the OOVm and produce useful results, we need to translate the qualitative relationships developed into quantitative expressions. Recently, work has begun to model mathematically each of the form systems and establish quantitative relationships between the systems. The mathematical models relate the quantitative function that each of the form systems performs to the static characteristics of the system (such as type, geometry, weight, cost). One can envision each of the forms as a "rubber" system -- given the functional characteristics required of the system, we can subject its model to mathematical variation and "stretch" the form into a "shape" that fulfills the function. For example, it may be appropriate to model the engine system so that its form characteristic "displacement" is a function of (in the mathematical sense) its functional characteristic "provide power." In this manner, if the function required is to "provide 150 HP," the engine system will be mathematically varied to have, say, "2.5 liters of displacement." Quantitative relationships between systems will allow mathematical aggregation of the sub-systems into an overall vehicle system. These numerical linkages between the mathematical models will also enable optimization to take place where spatial or functional conflicts between the systems exist or where some constraint is violated.

Unfortunately, development of quantitative relationships between mathematical models of the forms has proved to be difficult. Searches for pre-existing equations (e.g., in automotive handbooks, SAE standards, etc.) have turned up very few useful results. A promising solution to this dilemma, however, appears to be through the use of historical vehicle data to develop equations. A relational database of mechanical specifications, body dimensions, performance data, etc., (from sources such as the Environmental Protection Agency, Consumer Reports, and Car & Driver Magazine) for current passenger vehicles is presently being compiled. This database has been linked to a graphing software. This will enable us to make queries on the database and quickly see if a significant relationship exists between any parameters we desire to investigate. It is surmised that this effort will yield an abundance of useful equations which will form a

BASELINE PARAMETER TO FIRST-LEVEL FORM RELATIONSHIPS

-  STRONG RELATIONSHIP
 MODERATE RELATIONSHIP
 WEAK RELATIONSHIP

	TRANSPORTATION/ SUPPORT SYSTEM	POWER GENERATION/ TRANSFORMATION SYSTEM	ENTERTAINMENT/ DRIVER INTERFACE SYSTEM	COMPUTER CONTROL SYSTEM
NUMBER OF PEOPLE	◎	◎	◎	
NUMBER OF DOORS	◎		○	
ENGINE SIZE	○	◎	△	
CARGO CAPABILITY	◎	◎	○	
WHEEL BASE	◎	○	○	
DRIVE TYPE	◎	◎	△	
MATERIAL PERCENTAGE	◎	◎	△	
SILHOUETTE	◎	○	◎	
CLEARANCE	◎	○	○	

Figure 5.5

BASELINE PARAMETER TO SECOND-LEVEL FORM RELATIONSHIPS

- ◎ STRONG RELATIONSHIP
 ○ MODERATE RELATIONSHIP
 △ WEAK RELATIONSHIP

	TRANSPORTATION/ SUPPORT SYSTEM		POWER GENERATION/ TRANSFORMATION SYSTEM		ENTERTAINMENT/ DRIVER INTERFACE SYSTEM		COMPUTER CONTROL SYSTEM	
	STRUCTURAL SYSTEM	CHASSIS SYSTEM	POWER GENERATION (ENGINE) SYSTEM	POWER TRANSMISSION/ TRANSFORMATION SYSTEM	INTERIOR SYSTEM	EXTERIOR SYSTEM	INTEGRATION SYSTEM	EXTERNAL INTERFACE SYSTEM
NUMBER OF PEOPLE	◎	○	○	○	◎	◎		
NUMBER OF DOORS	◎		△		◎	◎		
ENGINE SIZE	◎	◎	◎	◎	○	○		
CARGO CAPABILITY	◎	○	○	○	○	○		
WHEEL BASE	◎	○	△	△	○	○		
DRIVE TYPE	○	◎	◎	◎	△	△		
MATERIAL PERCENTAGE	◎	◎	◎	◎	○	○		
SILHOUETTE	◎		△	△	○	◎		
CLEARANCE	◎	◎	○	○		○		

Figure 5.6

significant part of the OOVm. For example, a plot of vehicle weight versus vehicle length revealed a high correlation and a curve fit to the data provided a relationship between the function and form parameters (see Figure 5.7). This effort is also proving to be enlightening. For example, it was mentioned above that it might be appropriate to model horsepower as a function of engine displacement. However, a plot of the two parameters yielded a much poorer correlation than expected (see Figure 5.8). This leads us to look for possible combinations of form characteristics that affect this functional characteristic. Of course, we can use the qualitative relationships developed as a guide.

5.4 Summary

Work to this point has finalized the inputs to and outputs from the OOVm. Vehicle form description and decomposition has taken place, and the qualitative relationships among forms, functions, and baseline parameters have been investigated. Subsets of the input and output parameters have been chosen with which to study the OOVm before it is implemented for the entire vehicle. These subsets will allow us to study our methodology to determine if it is correct. Work has also begun to establish quantitative relationships among functions and forms using historical data.

What is needed next is to decompose form down to a level consistent with the functional decomposition. The relationships among the output variables and the vehicle forms as well as the spatial interactions among the forms need to be investigated. The development of qualitative equations relating forms and functions will be continued, and construction of a form relational database will begin. Using the relationships established, the process for function to form transformation and form synthesis and optimization will also be designed. As stated above, we will test the methodology developed using a subset of the OOVm.

VEHICLE WEIGHT vs. LENGTH

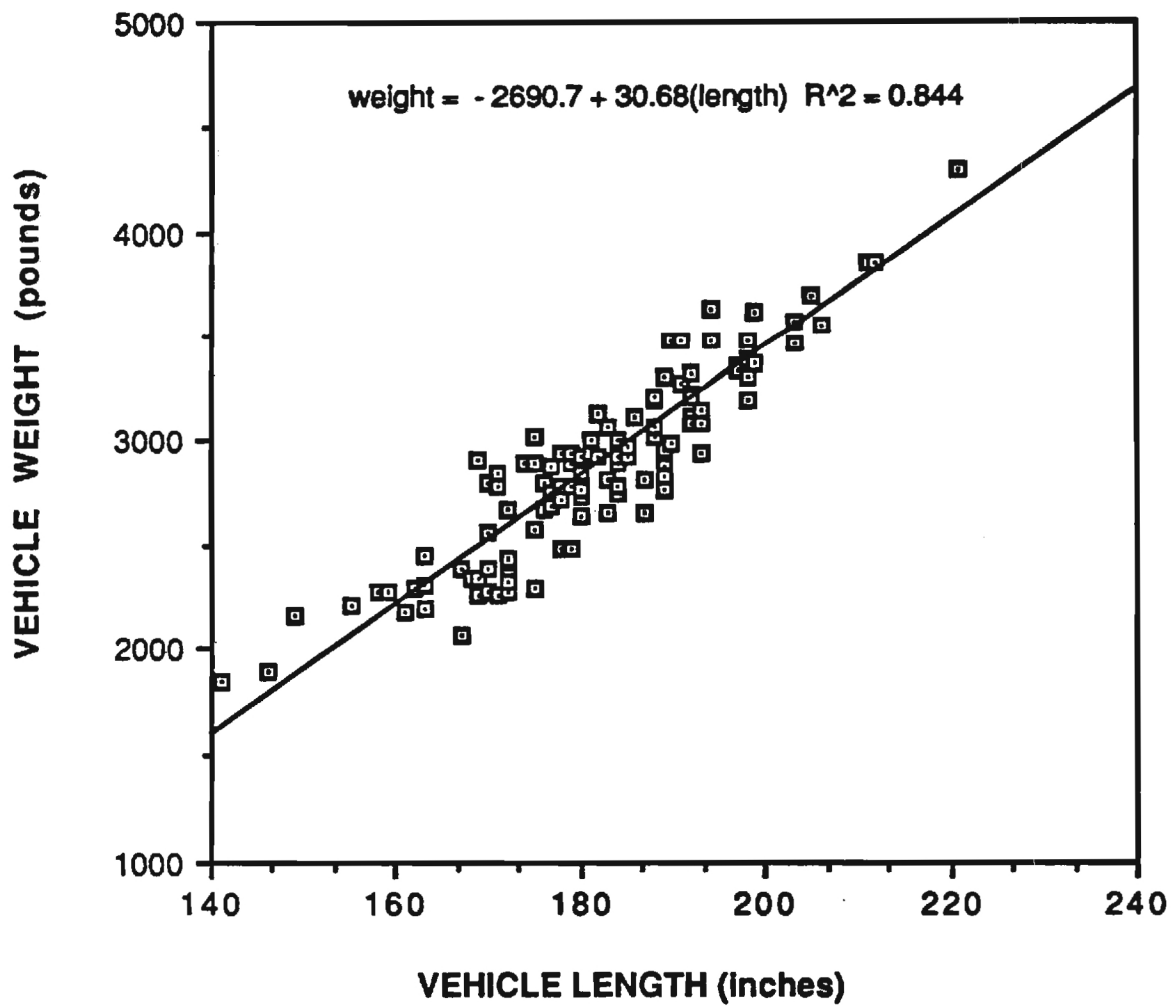


Figure 5.7

DISPLACEMENT vs. HORSEPOWER

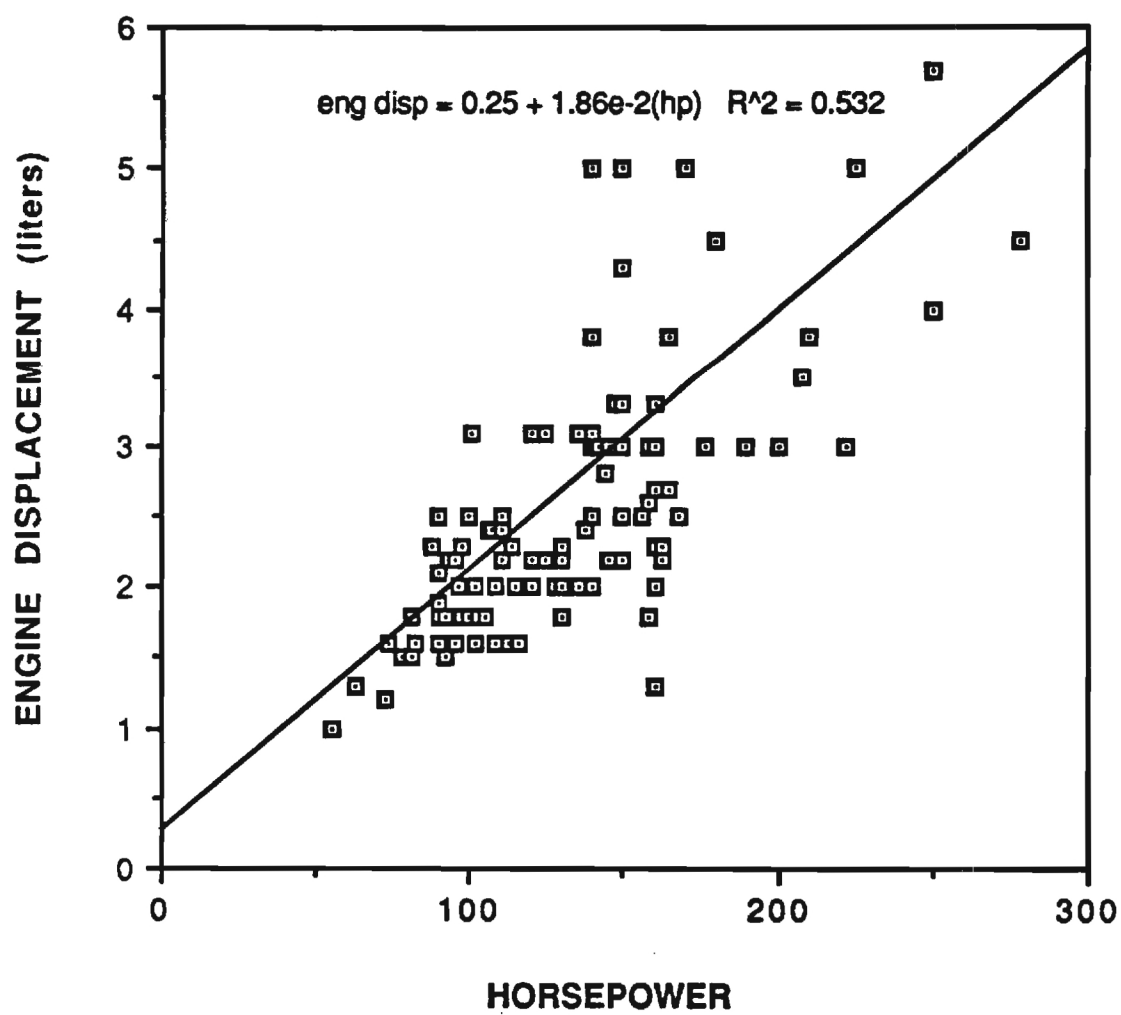


Figure 5.8

Chapter 6 A SAMPLE OF VOICE OF THE CUSTOMER/ FUNCTION/ FORM LINKAGES

6.1 Introduction

This chapter describes the design and implementation of a subsystem of the Object Oriented Vehicle Model (OOVM) [14], namely, a car engine. The subsystem was developed in order to test the methodology of the OOVM. This chapter describes the development of the system and the hardware and software platforms used.

6.2 Engine Subsystem

As stated above, the engine was used as a subsystem of the OOVM in order to test how the methodology will work for a smaller system. The system takes as its input the voice of the customer specific to the type of car desired. The inputs are the acceleration, weight, and cost of the car. These input-parameters are translated through a software called Hypercard into useful functions, which are then stored in Oracle (a relational database) as a goal-list.

Oracle contains the goal-lists, the form-lists, a list that describes what form fulfills what function, a list that describes what functions are needed by what forms to perform those functions, a list of equations related to the forms, a list of names of forms, and a list of names of functions. Because the model is kept simple, the number of possible forms is presently limited to 10.

The functions can be mapped by CLIPS (an expert system) into the form-lists. A form optimizer, not created within the sub-system, would then be used to select the best form from the choices in the table. The communication between CLIPS and Oracle is through an in-house developed C-program.

The output is the table of form-lists. These form-lists contain rubber forms which are moulded into the best configuration by the form-optimizer.

6.3 Software

6.3.1 The User-interface

Hypercard is an easy to use program. It can very easily set up a user interface for the car-system. Because the input parameters will be put in the database and handled as a range, the customer is asked to give a range in which the parameter should be. (i.e. the cargo-space should be somewhere between 50 and 75 liters).

Three different options could be used:

- typing to values into a field
- using slides on a scale
- dragging a picture into its right form.

The best way to do it on Hypercard is using slides. A field can show the numerical value that the slide represents. A qualitative value can give the user extra information he can use to get the right input. Also it is possible to make a drawing of the vehicle that shows the user what the car will look like with the given parameters. It takes a lot of time to make the pictures needed for that kind of output.

The interface made for the engine consists of a sidebar for each parameter. With each slide bar it is possible to give a range. The slidebars have an extra numerical and qualitative output.

6.3.2 The VOC -> Functions Translator.

The VOC -> functions translator can be programmed using four different programs:

- Hypercard script
- Hyper X, Hypercard's expert system
- a self made C-program
- CLIPS.

The data-stream can be handled directly and via Oracle, Hypercard's DataBase. Table 6.1 gives a table in which the interfaces between all the systems are described.

Table 6.1 Data-stream / Program Interfaces

Direct Data-stream

- HC and HC : Hypercard script can directly access other cards and stacks in Hypercard
- HC and HX : Hyper X is a hypercard stack so see HC and HC
- HC and C-Pr: It is possible to make an XCMD in Hypercard that can read information into a buffer and start up a C-program that can read from that buffer. It is NOT possible to read data from Hypercard stacks when not using the buffer. So all the information necessary should be passed through the buffer. (succeeded in starting up the program, not succeeded in transferring data)
- HC and CLIPS : CLIPS is a C-program so see HC and C-pr

Information via ORACLE

To pass information from ORACLE to and from the different programs the following ways are used.

- HC to ORACLE : The XCMD execsql can be used to give SQL-commands to talk to ORACLE. To be able to use the XCMD rescopy should be used to copy the resource code from the ORACLE system-stack to the self made interface-stack.

- ORACLE to HC : See HC to ORACLE

- ORACLE to HX : See HC to ORACLE

- ORACLE to C-pr : When using the pro C* precompiler it is possible to write SQL commands in a C-program. This can be used to read/write data from/to ORACLE by the C-program.

- ORACLE to CLIPS : CLIPS is a C-program so pro C* is also used here. There are two ways to go :

- first read all the information from ORACLE and assert them as facts into the fact-list of CLIPS and then start the consultation.

- write user-functions for CLIPS so the data can be read into the fact-list during a consultation.

By using the first method the consultation will be quicker, but to start up the consultation all data that is not used is also read into the fact-list. A second problem could be memory, because all the information in ORACLE is read into the memory of the computer. So overall the second method is better.

To pass data from the different programs to ORACLE the programmer can use the same programs as described above.

When the mapping is very easy, a Hypercard script should be used, as it is the simplest. If the translator gets bigger and more complicated a C-program is more helpful. When the problem is so complicated that it is necessary to write an inference engine, CLIPS should be used. Hyper X is not the easiest expert system to work with and is not suitable for an easy solution.

6.3.3 The Function-optimizer.

The function-optimizer is not implemented in the engine system. Because the model of the engine was simple a function-optimization was not necessary. When the system is expanded, it will be necessary to include it. The input for the function-optimizer is a goal_list with functions and the

ranges of the function attributes for those functions. The output is a goal_list with functions and ranges that are optimized. This goal_list is put into ORACLE.

The function-optimizer is normally more complicated than the VOC-function translator. Therefore a C-program or CLIPS is more suitable to make it. The necessary program interfaces are described in Table 6.1.

6.3.4 The Function -> Form Mapper

The mapper that is implemented now is written in a C-program. It uses the pro C* [10] commands to communicate with ORACLE.

Hypercard or a C-program puts the first goal_list into ORACLE. The form_list that belongs to this list is empty. The input for the mapper is as follows:

- the goal_list
- an empty form_list
- information on which form can fulfill what functions and what functions it needs to be fulfilled when performing those functions.

The output is a list of formlists of which each of the lists can perform the functions that were in the original goal_list. The steps followed by the mappers are as follows:

- 1) The mapper reads the first goal_list and starts looking for forms that can fulfill the first function in the goal_list. These forms are put into a select_list of forms.
- 2) The select_list is minimized. This will be done by an intelligent minimizer, the engineer, or a combination of those two.
- 3) The first form in the select_list is checked to see if it is usable.
- 4) If so, a new form_list is made adding the new form to the current form_list.
- 5) A new goal_list is made taking all the functions fulfilled by that form out of the goal_list.
- 6) Then the functions necessary for that form to operate are added to the new goal_list.
- 7) The steps 2 to 5 are repeated for every form in the select_list.
- 8) The steps 1 to 6 are repeated for every goal_list.

An example of this process is shown in Table 6.2.

Table 6.2 The Working of the Mapper

Example on the mapper:

The goal_list 0 is put into ORACLE by Hypercard or a C-program.

Functions are numbers, Forms are letters.

form A fulfill(1, 2) necessary (3)

form B fulfill(1) necessary ()

form C fulfill(2) necessary()

form D fulfill(3) necessary()

goal_list 0 : (1, 2)

form_list 0 : ()

select_list (A, B)

A :

form_list 1 : (A)

goal_list 1' : ()

goal_list 1 : (3)

B :

form_list 2 : (B)

goal_list 2 : (2)

goal_list 1 : (2)

form_list 1 : (A)

select_list (D)

D :

form_list 3 : (A, D)

goal_list 3 : ()

goal_list 2 : (B)

form_list 2 : (2)

select_list (C)

C :

form_list 4 : (B, C)

goal_list 4 : ()

The problems that occur with the macintosh computers are memory management problems. An exact description is given in section 6.A.2.

In addition to step 2 : The minimizer

To optimize the mapping process, not all the possible forms should be put in a list and given as output. If, for instance, two different forms can perform the same function and if they also need the same functions to be fulfilled to operate the one should be taken that fits the best in the profile of the car that the customer has given. The minimization can be done by an expert or by an expert computer system. An additional minimization can be done at the end of the mapping process or just before the form_optimizer.

An output session of the CLIPS mapper is shown Table 6.3.

Table 6.3: CLIPS Mapping Output

```
CLIPS> (reset)
CLIPS>

(load "mapper")
Compiling rule: init +j
Compiling rule: form-found +j+j+j+j+j+j+j
Compiling rule: print-it =j+j+j
Compiling rule: delete-wait +j+j+j+j+j
Compiling rule: get-rest =j+j+j+j
Compiling rule: stop-rest =j=j+j+j+j
Compiling rule: print-stop +j+j
CLIPS>

(reset)
CLIPS> (run)

The list is  cylinder carburetor spark-plug cooling air-cleaner fuel-
cleaner battery environment

The list is  cylinder injector cooling air-cleaner fuel-cleaner environment

The form injector fulfills function mixed-air-fuel
that already is fulfilled
The list of functions is : cooling clean-air clean-fuel
The list of forms is   : cylinder carburetor injector

35 rules fired
Run time is 5.58325195 seconds
CLIPS> (dribble-off)
```

6.3.5 The Form-optimizer

The output of the mapper are rubber forms with the equations that describe these forms. This information is passed into an IBM computer. Smalltalk and Demaid run on this computer and they perform the form-optimization. The transfer between the two computers can be done by an on-line link or by using floppies.

6.4 Oracle Database

ORACLE is used as the Hypercard's database. It is a relational database. The tables in the database are setup in third normal form. The tables are as follows:

FORMS: - id
 - name

Used to link form-names to form-id numbers.

FUNCTIONS: - id
 - name
 - unit

Used to link function-names to function id-numbers. Unit gives the unit(s) in which the function is measured. The tables **FULFILL**, **NECESSARY** and **GOAL_LIST** don't give units, only numbers.

FULFILL: - id
 - form_id
 - function_id
 - bottom-value
 - top_value

Used to describe which forms can **FULFILL** what functions in what range.

NECESSARY: - id
 - form_id
 - function_id
 - bottom-value
 - top_value

Used to describe which functions in what range are needed for a form to fulfill the functions described in the table **FULFILL**.

EQUATIONS_LIST: - id
 - form_id
 - equation

Used to give the equations that belong to a form.

ERROR_MESSAGES: - id
 - message

FORM_LIST: - id
 - list_num
 - form_id

Used to store form_lists. List_num gives the number of the list. Form_id gives the form that is in that list.

GOAL_LIST: - id
 - list_num
 - function-id
 - bottom-value
 - top_value

Used to store goal_lists. List_num gives the number of the list. Function_id gives the function that is in that list. Bottom-value and top_value give the boundaries of the range in which we want the function to be fulfilled.

The last two tables work as follows :

If list #1 is '1 : (A, B, C)' and list #2 is '2 : (A, B)' then the entries in the table are as follows:

id	list_num	form
1	1	A
2	1	B
3	1	C
4	2	A
5	2	B

6.5 Conclusions

The Macintosh computer does not efficiently run a big system like the implementation of the OOVm. The problems that occurred were mainly memory management problems.

The engine is a good example to work with because it contains many of the problems one can find in designing an OOVm and is an easy model to make.

Hypercard is very good to use as a development environment for a user-interface.

As an expert system Hyper X is not as easy to use as CLIPS. Therefore everything is written in CLIPS.

Using the mapper the output will be a list of lists of all combinations of form that, when combined can fulfill the goal_list 0.

Both the C-program and CLIPS are usable as an environment to program a mapper. The C-program will be faster.

6.6 References

- [1] Obert, Edward F., *Internal Combustion Engines*, Scranton: International Textbook, Co. (1986).
- [2] *Practical Limits of Efficiency of Engines*, London: Institution of Mechanical Engineers (1986).
- [4] *Aspects of Internal Combustion Engine Design*, Warrendale, PA: Society of Manufacturing Engineers (1982).
- [5] Ferguson, Colin R., *Internal Combustion Engines, Applied Thermoscience*, New York: Wiley (1986).
- [6] Ramos, J.I., *Internal Combustion Engine Modeling*, New York: Hemisphere Pub. Corp (1989).
- [7] Shafer, Dan, *Hypertalk programming (for Hypercard Version 1.2)*, Indianapolis: Hayden Books (1988).
- [8] Bond, Gary, *XCMD's for Hypercard*, Portland: MIS Press (1988).
- [9] Miller, Garth, *The Complementary Roles of Expert Systems & Database Management Systems in a Design for Manufacture Environment*, M.S. Thesis, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA (1990).
- [10] *Manual on ORACLE 1.2 for Macintosh Kit*, Oracle Corp., Redwood Shores, CA.
- [11] Giarratano, Joseph C., *Clips User's Guide (for Version 4.3)*, A.I Section, L.B. Johnson Space Flight Center, Houston, TX (1989).
- [12] *C for Yourself*, Microsoft Corp., Redmond, WA (1988).

[13] Waite, Mitchell, and Prata, Stephen, *C: Step by Step*, (1989).

[14] Colton, J., Craig, J., Fadel, G., Fulton, R., Heifetz, D., Lambright, J., LeBlanc, A., Preminger, J., "The Requirements for an Object Oriented Vehicle Model, Hierarchical Decomposition Design Methods for Automobiles", Interim Report to GM Systems Engineering, August 3, 1990.

6.A.1 Definitions

Form : A form is a physical part of a car (i.e. steering wheel, seat, gearbox).

Function : What a form can do (i.e. steer car, provide seating comfort, transmit power).

Voice of the customer : parameters of the car set by the customer.

Form_list : A list that contains the forms which, when combined, can perform the desired functions.

Goal_list : Goal_list 0 is the list of functions the user wants the car to perform. Goal_lists with higher numbers are subgoal set by the mapper.

6.A.2 Memory Problem using Macintosh's Think C and ORACLE.

The mapper is programmed in Lightspeed C(4.0). The program reads a goal_list out of ORACLE into a linked list in C. To make the linked list, memory is allocated for a field that can contain one function of the goal_list. An item of the goal_list is read from ORACLE and is written into the field. Then the field is linked into the linked list. This is done for all the items in the current goal_list. When the program gets bigger, the memory allocation function works OK. It gives a pointer to a memory field. When the program tries to write anything into the field, the computer quits the running process or even resets itself. The error is probably out of memory. The program however takes just about 60K of memory. Freeing memory by not using macbugs, the gatekeeper, the vaccine or the multifinder didn't work.

6.A.3 The Hard- and Software used

The system is implemented on a Macintosh IIx computer. The software available was Hypercard, HyperX, ORACLE, Lightspeed C, CLIPS and Microsoft WORD.

Chapter 7 A THEORETICAL APPROACH TO HIERARCHICAL DESIGN

7.1 Introduction

This chapter outlines a theoretical approach to hierarchical design which can provide the foundations of a process to seek the optimum design solution for a complex system. The basic concept envisioned is that there will be several objective functions, when combined appropriately, will define the merit of a design. For example, suitable combinations of three overall functions, Transport, Please, and Protect discussed in chapter 4, could represent the measures of goodness for a vehicle. An approach such as this can then provide a mechanism to implement major changes in the relative importance of Transport, Please, or Protect. Such an approach will require formalization of an optimum design approach, which is based on a multiple design objective function, $f_i(\mathbf{x})$. This approach can be denoted multiobjective optimization and is described and discussed here in the context of automotive vehicle applications.

7.2 Multiobjective Optimization and Design

Multiple figures of merit are applicable to virtually any design project. Multiobjective optimization problems can be posed as Pareto-optimality problems. Pareto-optimality refers to the situation where there are n figures of merit, which are functions of the design variables, $f_1(\mathbf{x}), \dots, f_n(\mathbf{x})$. A design, as characterized by the values for the design variables \mathbf{x} , is said to be Pareto-optimal if no one of the goals f_i can be improved without making some other goal f_j worse.

If all of the goals are formulated as minimizations, the Pareto-optimal designs can be found by minimizing a weighted sum of the individual goals, $\sum \omega_i f_i(\mathbf{x})$, where $\sum \omega_i = 1$ and each ω_i satisfies $0 \leq \omega_i \leq 1$.

In designing a vehicle system or subsystem, the values of the weights ω_i represent relative prioritizations of requirements by the customer. These relative prioritizations are rarely known precisely. Changes in market conditions may impact the prioritization of the design goals. The relative prioritizations are also subject to change as the product development team gains insight into the impact of the prioritizations on the design solution. Thus, in preparing to match design requirements with customers' prioritizations, it would be extremely useful to be able to identify families of design solutions corresponding to various values of the relative prioritizations.

Determining these families of design solutions would normally require varying the values for the weights ω_i and re-optimizing the design.

The cost of this procedure is high enough that such an approach is rarely, if ever, used in current design practice.

7.2.1 A New Technique Using Parameter Passing

This reoptimization can be avoided if the design problem can be decomposed into sub-problems where each of the design goals is assigned to a distinct subproblem.

A parameter passing scheme can be set up for such a decomposition which will generate all of the Pareto-optimal solutions (for all values of $\omega = (\omega_1, \dots, \omega_n)$). Only n design optimization problems must be solved; one for each objective. The solutions to these problems are then used to construct trade-off curves for all values of ω . The essential arguments in the proof of these results are summarized here. The proofs themselves can be found in [ROGAN & CRALLEY].

Proposition 1: Let \mathbf{x}^* be a point generated by the decision sequence. If all df/dx_i are 0 at \mathbf{x}^* , then \mathbf{x}^* is a Kuhn-Tucker-Karush point.

Proposition 2: Let the number of design variables in each decision element D_i equal or exceed the number of constraints in D_i which are independent and active at a point \mathbf{x}^* determined by a feasible, optimizing decision sequence. Then if \mathbf{x}^* is a Kuhn-Tucker-Karush point for the original problem P , all of the optimal sensitivity derivatives df/dx_i will be 0.

Proposition 3. $dF/dp(\omega) = \sum \omega_i df_i/dp$.

Propositions 1 and 2 are applied to parameter passing schemes for solving multiobjective design optimization problems by noting that the optimality conditions

$$dF/dp_i(\omega) = 0$$

for the approximate Pareto-optimization problem are the same as the convergence criteria for the solution of the exact Pareto-optimization problem by a parameter-passing scheme. The exact Pareto-optimization problem is solved, then, when these conditions are satisfied. Proposition 3 allows the conditions to be satisfied by varying the relative prioritizations, while maintaining the significance of those conditions in terms of optimal sensitivity derivatives. Thus, propositions 1, 2, and 3, establish the validity of techniques which solve the exact Pareto-optimization problem by varying the relative prioritizations in an approximate problem.

7.2.2 Application to an Automobile Configuration Problem

Providing volume is a design engineering problem that may be quite important in configuring an automobile. Methods for solving this problem are applicable to sizing the passenger, engine, and cargo compartments, as well as to sizing of containers such as fuel tanks (Figure 7.1). The problem can be described in terms of the baseline conceptual design parameters (identified in the main body of this progress report):

1. Number of people (rated)
3. Engine size
4. Cargo capability
5. Wheelbase
8. Silhouette
9. Ground clearance

In developing a design process to configure (select the shape and arrangement of) volumes for passengers, engine, cargo, fuel, and other containers, engineering theories and models describe how functions such as Transport, Protect, Please and Operate Safely are accomplished by specific elements of the vehicle form decomposition. The relationships represented by these models are linked to the form and function decompositions using QFD.

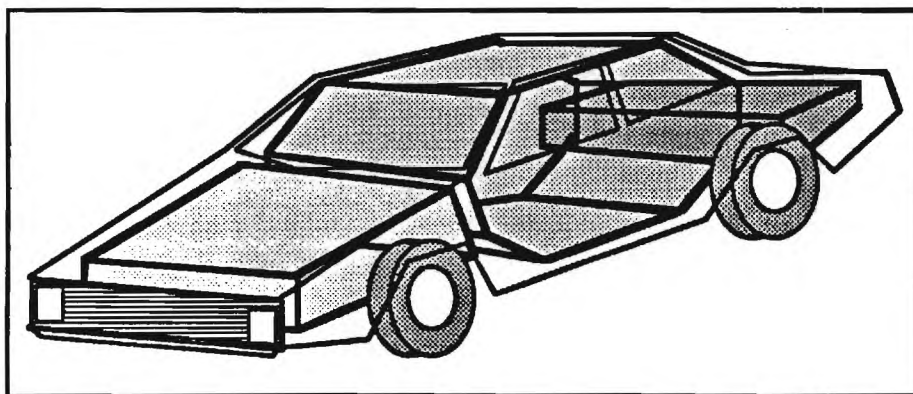


Figure 7.1. Cost/Capacity Problems in Configuring an Automobile.

This example is based on a volume configuration problem: it is required to find the dimensions (length, width, and height) of a rectangular box with surface area less than 6 ft.² ("relative materials cost") and volume greater than or equal to a target value of 10 ft.³ ("capacity"). In addition, the box cannot exceed 2 ft. in height.

The problem statement can be summarized as follows:

1. Choose length, l ; height, h ; and width, w of a box-shaped container.

2. Each dimension of the box-shaped container is between 0.5 ft. and 5 ft.
3. Capacity of the container, $f_1(l, w, h) = lwh$, is to be maximized, but not less than 10 ft³.
4. Relative cost of materials, $f_2(l, w, h) = 2(lw + wh + hl)$, is not to exceed 6.
5. Height of the container, h is to be less than 2 ft.

Note that alternative design concepts, attributes, and engineering theories and models have been specified in the statement of the design problem. Thus, this problem has been posed at an appropriate point in the system engineering process for the application of a design methodology. The information contained in the statement of this design problem can be represented graphically as a "design-in-process graph", Figure 7.2.

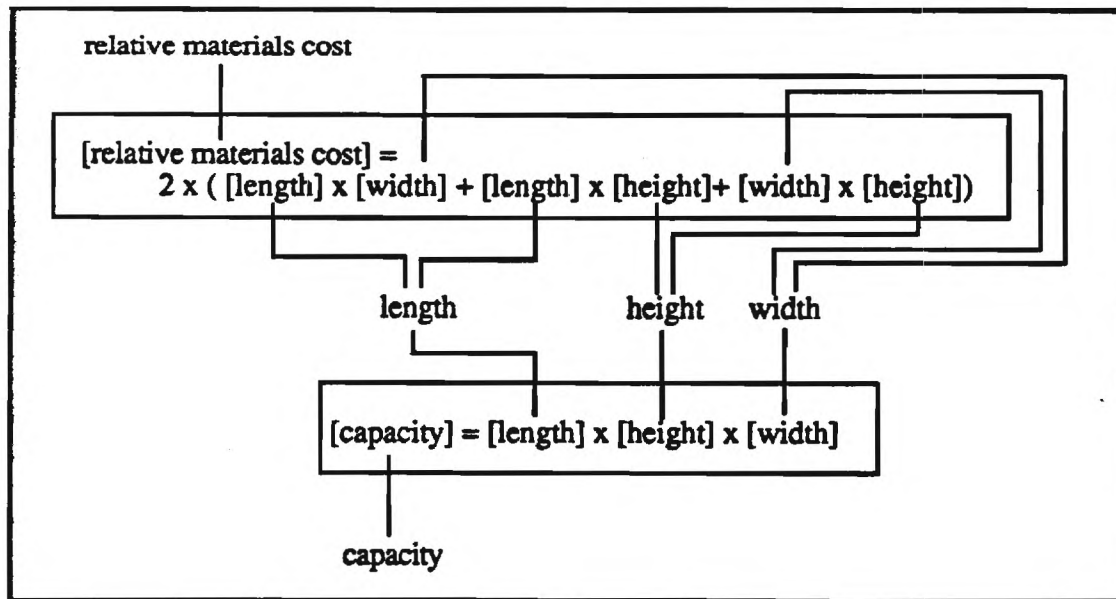


Figure 7.2. "Design-in-Process graph".

Design Methodologies for the Container Design Problem

A design methodology is a procedure for applying the engineering theories and models to determine values for attributes of the design, such as length, height, and width. As example design methodologies, consider the following:

Methodology A

1. Determine length, width, and height.

2. Apply

$$[\text{length}] \times [\text{height}] \times [\text{width}] = [\text{capacity}]$$

to determine capacity.

3. Apply

$$2 \times ([\text{length}] \times [\text{width}] + [\text{length}] \times [\text{height}] + [\text{width}] \times [\text{height}]) = [\text{relative materials cost}]$$

to determine cost.

Methodology B

1. Fix capacity, cost and height.

2. Solve

$$[\text{length}] \times [\text{height}] \times [\text{width}] = [\text{capacity}]$$

for length,

$$[\text{length}] = [\text{capacity}] / ([\text{height}] \times [\text{width}]).$$

Substitute this relationship into the equation

$$2 \times ([\text{length}] \times [\text{width}] + [\text{length}] \times [\text{height}] + [\text{width}] \times [\text{height}]) = [\text{relative materials cost}]$$

and solve

$$2 \times ([\text{capacity}] / [\text{height}] + [\text{capacity}] / [\text{width}] + [\text{width}] \times [\text{height}]) = [\text{relative materials cost}]$$

for width:

$$w = \{ - (c_1/h - c_2/2) \pm [(c_1/h - c_2/2)^2 - 4hc_1]^{1/2} \} / (2h)$$

where w = width, c_1 = capacity, c_2 = cost, and h = height.

Determine width from cost, height, and capacity.

3. Use the equation

$$[\text{length}] = [\text{capacity}] / ([\text{height}] \times [\text{width}])$$

to find the length.

Methodology C

Solve the following optimization problem:
maximize:

$$c_1 = lwh$$

subject to:

$$2(lw + wh + lh) \leq c_2$$

$$c_2 = 6$$

$$c_2, l, w, h \geq e > 0, h \leq 2.$$

The design vector decomposition $\mathbf{x}_1 = (c_2)$, $\mathbf{x}_2 = (l, w, h)$ can be used for this problem. The decision elements are then

C_1 :

satisfy: $c_2 = 6$

$$2(lw + wh + lh) \leq c_2$$

design variables: c_2

fixed parameters: l, w, h

and

C_2 :

maximize:

$$c_1 = lwh$$

subject to:

$$2(lw + wh + lh) \leq c_2$$

$$l, w, h \geq e > 0, h \leq 2$$

design variables: c_1, l, w, h

fixed parameters: c_2

Since there is a unique objective function for the problem addressed by Methodology C, c_1 , weighting factors need not be defined.

Methodology D

Solve the following optimization problem:
maximize:

$$c_2 = 2(lw + wh + lh)$$

subject to:

$$lwh \geq c_1$$

$$c_1 = 10$$

$$c_1, l, w, h \geq e > 0, h \leq 2.$$

The design vector decomposition $\mathbf{x}_1 = (c_1)$, $\mathbf{x}_2 = (l, w, h)$ can be used for this problem. The decision elements are

D_1 :

satisfy: $c_1 = 10$
 $lwh \geq c_1$
 design variables: c_1
 fixed parameters: l, w, h

and

D_2 :

maximize:
 $c_2 = 2(lw + wh + lh)$
 subject to:
 $lwh \geq c_1$
 $l, w, h \geq e > 0, h \leq 2$

design variables: c_2, l, w, h
 fixed parameters: c_1

Again, there is a unique objective function for the optimization problem addressed by Methodology D, c_2 , making weighting factors unnecessary.

Methodology E

Solve the following optimization problem:
 minimize:

$$\omega_1 [c_1/10 - 1]^2 + \omega_2 [c_2/6 - 1]^2$$

subject to:

$$\begin{aligned} lwh &\geq c_1 \\ 2(lw + lh + wh) &\leq c_2 \\ \omega_1 + \omega_2 &= 1 \\ c_1, c_2, l, w, h &\geq e > 0, h \leq 2 \\ \omega_1, \omega_2 &\geq 0 \end{aligned}$$

The design vector decomposition $\mathbf{x}_1 = (\omega_1, \omega_2)$, $\mathbf{x}_2 = (c_1, c_2, l, w, h)$ can be used for this problem. Decision elements are

E_1 :

minimize:

$$\omega_1 [c_1/10 - 1]^2 + \omega_2 [c_2/6 - 1]^2$$

subject to:

$$\begin{aligned} \omega_1 + \omega_2 &= 1 \\ \omega_1, \omega_2 &\geq 0 \\ c_1, c_2 &\text{ fixed parameters} \end{aligned}$$

and

E_2 :

minimize:

$$\omega_1 [c_1/10 - 1]^2 + \omega_2 [c_2/6 - 1]^2$$

subject to:

$$\begin{aligned}
 lwh &\geq c_1 \\
 2(lw + lh + wh) &\leq c_2 \\
 c_1, c_2, l, w, h &\geq e > 0, \quad h \leq 2 \\
 \omega_1, \omega_2 &\text{ fixed parameters}
 \end{aligned}$$

7.2.3 Analysis of Design Methodologies

Each of these methodologies can be represented as a directed graph. In the directed graph representation, the decision elements of the methodology are the nodes or vertices. The sequence of design decisions embodied in the methodology is represented by arrows (directed edges) connecting the decision elements.

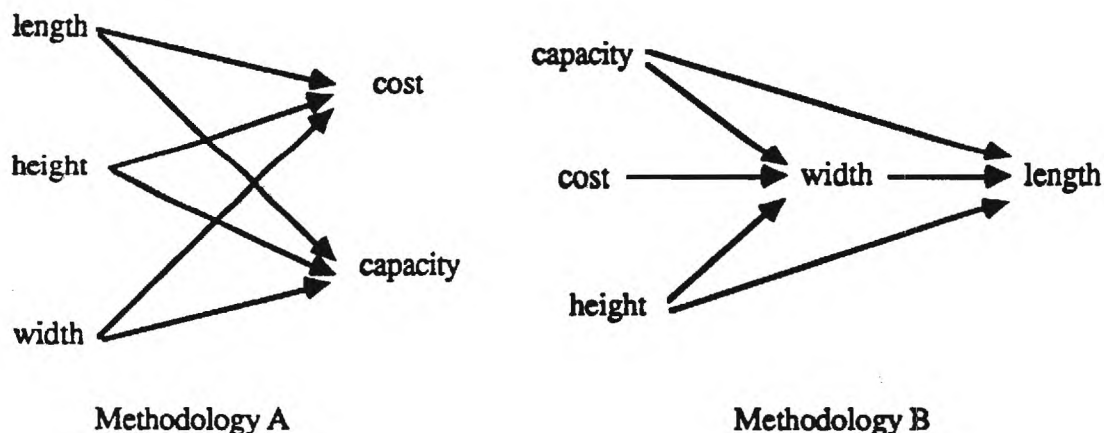


Figure 7.3. Directed Graph Representations of Design Methodologies A and B.

The information content of these directed graphs is sufficient for the application of decision structuring techniques (such as Interpretive Structural Modelling and the Design Structure System) which are based on connectivity information (see [ROGAN & CRALLEY] for a more in-depth discussion of these techniques). In fact, these graphs are probably far too simple to produce interesting results using the decision structuring techniques. It is clear that directed graphs can be used to identify obvious problems with the methodologies. For example, from the directed graph of methodology A (Figure 7.3), it is evident that unless the correct values for the attributes length, width, and height are already known, it is practically impossible to meet the cost and capacity requirements without iterating the decision-making process. Methodology A does not specify such an iterative strategy, so methodology A is rejected as too risky.

However, even for these simple design methodologies, the directed graph representation is not adequate to accurately evaluate their suitability as an approach to solving the container design problem. This is evident in the analysis of methodology B. The directed graph of methodology B (Figure

7.3) indicates that it should be possible to determine length and width, the design variables, from cost, capacity, and height, the requirements. This appears to correspond precisely to the container design problem. However, substituting cost = 6, capacity = 10, and height = 2, into

$$w = \{-(c_1/h - c_2/2) \pm [(c_1/h - c_2/2)^2 - 4hc_1]^{1/2}\}/(2h)$$

the solution $(c_1/h - c_2/2)^2 - 4hc_1 = -76 < 0$ is found, so there can be no real solutions for the width, w . This indicates that the initial requirements are not feasible. No indication of this problem can be seen in the directed graph representation. In fact, the directed graph indicates that methodology B is well-matched to the container design problem. It is clearly not possible to determine whether a design methodology will lead to feasible designs using only the information in the directed graph representation. Analytical concepts of optimization theory are needed to bring feasibility into the evaluation. The connectivity information contained in the directed graph representation is simply not adequate for this task.

Interpreting some of the requirements identified in the container design problem as goals makes it possible to model them, in the context of optimization theory, as objective functions. Requirements that cannot be relaxed are modelled in optimization theory as constraints. Taking advantage of this, optimization theory, specifically convergence theory, is applied to assess the capability of the remaining design methodologies to solve the problem. It is emphasized that this application of optimization theory to assess a design methodology is distinct from the application of design optimization methods, or more specifically numerical optimization, as a part of a particular design methodology.

Optimization theory is applied to evaluate the suitability of a design methodology to meet a set of requirements by considering each step in the design methodology to be a decision element. In the context of optimization theory, decision elements are optimization or feasibility subproblems. These subproblems are related to one another by engineering theories and models linking design attributes in distinct decision elements. In evaluating the suitability of a design methodology to meet requirements, engineering theories are analyzed to determine the monotonicity of the relationships among attributes implied by the engineering theories and models. Thus, for example, if width is increased with length and height fixed, an increase in capacity is required to satisfy the engineering theory

$$c_1 = lwh.$$

Thus, capacity is monotonically increasing with length when the "capacity" engineering theory is enforced.

In a design methodology with multiple steps, the values of design attributes will be changed as decisions are made. The monotonicity information can be used to determine whether these decisions will

adversely affect the feasibility of subsequent decisions. The monotonicity information can also be used to assess the overall progress of the decision-making sequence toward an optimal, or alternatively, toward a Pareto-optimal (balanced) design. Convergence of a methodology to a feasible design and progress toward a balanced or optimal design can be ensured by imposing certain criteria on the sequence of decision elements. The simplest approach is to allow decision element D_i to be sequenced before decision element D_j only if the choices for values of design attributes in decision element D_i will not adversely affect feasibility or optimality of decision element D_j . Of course, such a sequence may not be possible to realize in practice. An alternative approach is then to constrain prior decision elements so that subsequent decision elements have feasible solutions.

To illustrate these ideas, once again consider methodology B. In methodology B, choices for height, capacity, and cost are distinct decision elements that are sequenced before a choice is made for the value of the width design attribute. Choice of the width attribute is in fact constrained by $w \geq 0$. Clearly, it is possible to choose values for the height, cost, and capacity attributes that make the width decision element infeasible. (for example, height = 2, cost = 6 and capacity = 10). Thus, using the concept that prior decisions should not adversely affect feasibility of subsequent decisions, one of the limitations of methodology B has been correctly identified. Methodology B has additional limitations. One of these limitations is the fact that methodology B does not provide any means to continually improve the design in terms of cost and capacity goals throughout the design process. To evaluate methodologies attempting to accomplish such improvements, it is necessary to consider optimality, or equivalently, Pareto-optimal balance, in addition to feasibility.

Methodologies C and D represent different approaches to the design optimization problem. In methodology C, if C_2 is solved before C_1 , the constraint $c_2 = 6$ may not be satisfied. The sequence $\{C_1, C_2\}$ is feasible: once the cost is fixed in C_1 , the optimization problem C_2 has a feasible solution for any positive value of cost. The analogy that methodology D is feasible if the capacity is positive is valid.

Do methodologies C and D lead to optimal or balanced designs? The answer to this question depends on who defines optimality. Optimality must be defined by the customer's needs. Thus, methodologies C and D can be optimizing only when they match the customer's ranking of the cost and capacity requirements. Methodology C provides the maximum capacity meeting the cost requirement, and methodology D delivers the minimum cost to meet the capacity requirement. This fact is reflected in the sequence of design decisions. In fact, the customer may need to balance cost and capacity in some sense. Neither methodology C or methodology D can address this balancing problem. Thus, methodologies C and D must be rejected. These methodologies cannot produce designs balancing cost and capacity.

Considerable additional complexity is required to fully address this problem of balanced design, as is illustrated by methodology E. Methodology E calls for a separate requirements ranking decision element (decision element E_1) in which relative weights for the cost and capacity goals are determined. Incorporation of this decision element ensures that methodology E will deliver balanced designs. In the second decision element in methodology E, decision element E_2 , values of cost and capacity goals are determined, resulting in a feasible solution of the Pareto-optimization problem.

The optimization problem in methodology E is stated so that a solution can always be found: even though the cost or capacity goals may not be met, the design will balance the degree to which those goals are achieved, with relative priorities determined by the weighting factors. Thus, the statement of methodology E ensures feasibility in this restricted sense. Thus, methodology E is well-matched to the container design problem. Unfortunately, the price exacted for this suitability is too high: the customer must be able to rank cost and capacity without the benefit of information about the design relationship between achievable values of cost and capacity. This ranking represents a question of comparable difficulty to the design problem itself.

Perhaps a more practical approach is to generate the cost/capacity curve and relate this curve to the relative prioritizations. This information can then be used in a requirements negotiation. Information to support requirements negotiation is most valuable before the design is completely specified. Once agreement is reached concerning goals and their priorities, the design of the water storage tank can be finalized. Methodology E is not well suited to this expanded problem. Using methodology E to generate the cost/capacity curve would involve varying the relative prioritizations, executing a reoptimization of decision element E_2 for each set of priorities. Thus, many containers would have to be designed before negotiations of the goals for the container cost and capacity could be concluded with the customer. In more complex design problems, each reoptimization, in itself, may involve the execution of a complete design methodology in a decision element such as E_2 .

7.2.4 A Design Methodology to Support Requirements Trade-Offs

An approach for efficiently generating families of Pareto-optimal solutions can be developed by further extending the application of optimization theory to design methodologies consisting of separate decision elements. In an optimization-based theory of design methodologies, the following question can be posed: how can information to support requirements negotiation be generated by executing relatively simple design methodologies, comparable to methodology C or D? This question is answered by a method based on propositions 1, 2 and 3.

Consider the following methodology for developing design information to support requirements negotiation:

Methodology F

Solve the following optimization problem (the same problem as solved by methodology E):

minimize:

$$\omega_1 [c_1/10 - 1]^2 + \omega_2 [c_2/6 - 1]^2$$

subject to:

$$\begin{aligned} lwh &\geq c_1 \\ 2(lw + lh + wh) &\leq c_2 \\ \omega_1 + \omega_2 &= 1 \\ c_1, c_2, l, w, h &\geq e > 0, \quad h \leq 2 \\ \omega_1, \omega_2 &\geq 0 \end{aligned}$$

The design vector decomposition $\mathbf{x}_1 = (c_1)$, $\mathbf{x}_2 = (c_2)$, $\mathbf{x}_3 = (\omega_1, \omega_2, l, w, h)$ can be used for this problem. Decision elements are as follows.

F_1 :

minimize:

$$f_1 = (c_1/10 - 1)^2$$

subject to:

$$c_1 - lwh \leq 0$$

design variables: c_1

l, w, h fixed parameters

and

F_2 :

minimize:

$$f_2 = (c_2/6 - 1)^2$$

subject to:

$$2(lw + wh + lh) - c_2 \leq 0$$

design variable: c_2

l, w, h fixed parameters

F_3 :

minimize:

$$F = \omega_1 f_1^{\text{opt}}(l, w, h) + \omega_2 f_2^{\text{opt}}(l, w, h)$$

subject to:

$$\begin{aligned} h - 2 &\leq 0 \\ \omega_1 + \omega_2 &= 1 \\ \omega_1, \omega_2 &\geq 0 \end{aligned}$$

where the design variables are now: $\omega_1, \omega_2, l, w, h$. f_1^{opt} and f_2^{opt} are the optimal values of the objective functions for subproblems F_1 and F_2 , respectively, with l, w , and h fixed.

Although methodology F is similar in some ways to methodology E, there is an important difference in that the multiple objective functions are assigned to different subproblems. Although the sequences of design decisions which make sense for methodology F are somewhat restricted (F_1 and F_2 must be executed before F_3 in order to define f_1^{opt} and f_2^{opt}), methodology F provides a very efficient technique for constructing the cost-capacity curve as a function of the requirements priorities.

The basic concept is that the optimality conditions for subproblem F_3 follow directly from the solutions to subproblems F_1 and F_2 . Thus, it is not necessary to solve F_3 . The formulation of F_3 is such that the optimality conditions for F_3 are the same as the optimality conditions for the undecomposed optimization problem addressed by both methodology E and methodology F. Thus, the particular decomposition strategy used in methodology F solves the Pareto-optimization problem indirectly, using the solutions to two suboptimization problems (subproblems F_1 and F_2). Although the difference between this solution technique and methodology E is inconsequential for the simple water storage tank design problem, methodology F can be applied to much more complex design problems with a few relatively simple extensions.

Details of the solution for methodology F are given in section 7.A.1. Optimality conditions for sub-problem F_3 are

$$\begin{aligned}\partial F / \partial p_1 &= \omega_1 D_{p_1} f_1^{opt} + \omega_2 D_{p_1} f_2^{opt} = 0 \\ \partial F / \partial p_2 &= \omega_1 D_{p_2} f_1^{opt} + \omega_2 D_{p_2} f_2^{opt} = 0 \\ \partial F / \partial p_3 + 1 &= \omega_1 D_{p_3} f_1^{opt} + \omega_2 D_{p_3} f_2^{opt} + 1 = 0.\end{aligned}$$

The key to methodology F is to fix the optimal sensitivity derivatives and regard these equations as determining values for ω_1 and ω_2 that correspond to those values of the optimal sensitivity derivatives. Sequential parameter passing schemes of the type exemplified by methodology F converge to the optimal solution of a Pareto-optimization problem such as the problem approximated by F_3 . The benefit accruing from this approach is that the entire family of Pareto-optimal solutions (corresponding to different values for the ω_i 's) is determined using only

- Information about the solution to an optimization problem corresponding to each of the objective functions, and
- Partial derivatives.

The alternative approach (methodology E) would require the solution of a separate optimization problem for each combination of values of the ω_i 's.

Continuing with the solution of the optimality conditions for subproblem F_3 , there are 5 unknowns:

ω_1 , ω_2 , l , w , and h ,

and 4 independent equations (three of which are nonlinear):

$$\omega_1 + \omega_2 = 1$$

(3 optimality conditions).

Thus, the entire family of Pareto-optimal solutions can be characterized by varying one of the parameters. One way to do this is to solve:

$$\begin{aligned} \partial F / \partial p_1 &= \omega_1 D_{p_1} f_1^{\text{opt}} + \omega_2 D_{p_1} f_2^{\text{opt}} = 0 \\ &= \omega_1 (1/5)(1 - c_1/10)(-wh) + \omega_2 (1/3)(c_2/6 - 1)(w+h) \end{aligned}$$

to obtain

$$\omega_1 = (1/3)(c_2/6 - 1)(w+h) / [(1/3)(c_2/6 - 1)(w+h) + (1/5)(1 - c_1/10)(wh)].$$

Apply the remaining two optimality conditions to determine that $l = w$ and $h = \min(w, 2)$. To present the results, the engineering theories and models relating cost and capacity to l , w , and h are reintroduced. These determine the achievable values for cost and capacity. The variables c_1 and c_2 of subproblems F_1 and F_2 refer to the cost and capacity constraints. The difference between these constraints and the achievable values is, of course, central to the whole point of requirements negotiation.

Vary w to determine cost, capacity and w_1 . The relationship between achievable capacity, actual cost, and requirements prioritization, w_1 obtained in this way is plotted in Figure 7.4. Since the cost and capacity requirements cannot both be met, the customer must accept some loss of capacity, cost, or both. The magnitude of the loss depends on the relative prioritization given to achieving each of the goals. The loss in capacity is defined as

$$\text{capacity goal} - \text{achieved capacity}$$

and represents the distance to the desired capacity of 10 ft³. The loss in cost is defined as

$$\text{actual cost} - \text{cost goal},$$

adopting the convention that losses are positive.

This information can then be used to negotiate values for the relative prioritizations ω_1 and $\omega_2 = 1 - \omega_1$. Once these values are fixed, the optimal values for the remaining design parameters are also determined.

In most cases, where the problem posed in methodology E cannot be solved explicitly, this procedure will be much more efficient than methodology E. This is a consequence of the fact that the optimization problem in methodology E must be solved a large number of times, one solution for each value of ω_1 , to determine the effect of ω_1 and ω_2 on cost/capacity relationship, while methodology F can generate the entire cost/capacity relationship from the solutions of a few optimization problems, one for each of the multiple objective functions.

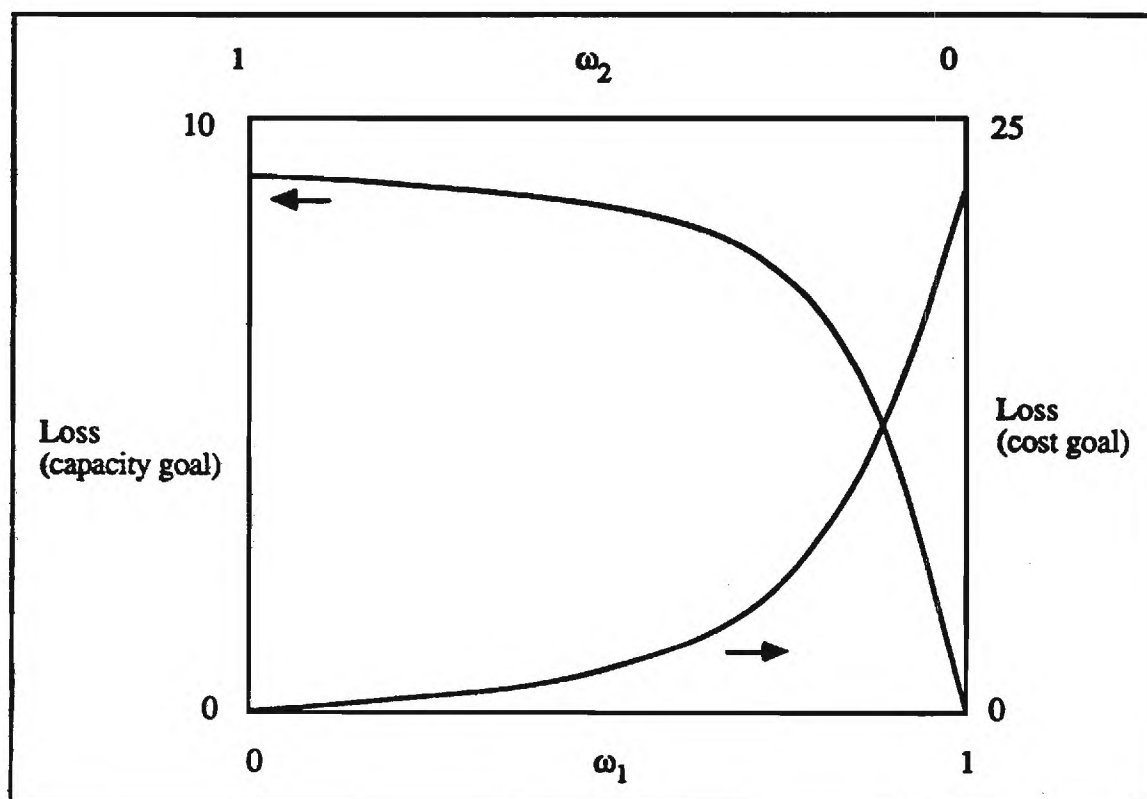


Figure 7.4. "Loss" of Cost and Capacity Goals vs. Requirements Priorities for a Water Storage Tank

7.2.5 Hierarchical Design using Optimizing Sequences of Decisions

As developed in the automobile configuration example, the assessment of a design methodology depends on three closely related results in optimization theory, propositions 1, 2, and 3. The first two propositions relate convergence of a large optimization problem to sensitivities of the solutions to individual decision elements appearing in the decision sequence defined by the design methodology. The statement of the large design optimization problem encompasses all of the attributes, goals, and the constraints imposed by requirements through engineering theories and models that are employed in the design methodology. This result is valuable, because a given design problem may have a statement as an optimization problem, but solution of the large problem may be impractical. The purpose of the analysis is to determine whether or not a given hierarchical decomposition and design methodology associated with it produce solutions to an optimization problem.

The second result extends the validity of this approach to include the solution of Pareto-optimization problems where the multiple objectives may be assigned to different decision elements. Pareto-optimal designs are

characterized by the statement that no one of a set of multiple objectives can be improved without making some other objective worse. Pareto-optimality thus equates balanced design to a form of optimal design. The extension of the methodology to handle Pareto-optimal solutions can be applied to establish convergence of a design methodology to a balanced design. The concept of balanced design broadens the range of problems that can be formulated to include situations where conflicting requirements may mutually exclude any solution, so that a balanced compromise must be drawn.

These results prove convergence of a sequence of design decisions to a balanced, feasible design. Criteria for convergence outlined in this section of the report were applied to assess alternative design methodologies for the simple automobile configuration example. Problems were found with all but one of the methodologies. An alternative methodology for balanced design was defined and applied to the container design problem. This method uses the capability to place different objectives of a multi-objective problem into different decision elements.

Once an engineering design concept has been chosen for synthesis, the design decision-making problem can be stated as a Pareto-optimization problem:

$$\text{minimize: } \sum \omega_i f_i(\mathbf{x})$$

$$\begin{aligned} \text{Subject to: } & g(\mathbf{x}) \leq 0 \\ & h(\mathbf{x}) = 0 \end{aligned}$$

where \mathbf{x} is a vector of design variables, $f_i(\mathbf{x})$ are design goals or objectives, and $g(\mathbf{x})$ and $h(\mathbf{x})$ are vector functions of the vector \mathbf{x} which represent requirements or constraints. The ω_i are relative prioritizations of the design goals or objectives. The importance of this problem statement is that it provides a theoretical framework for the study of the design process.

The principal tools for applying optimization theory to study the design process are:

- optimal sensitivity derivatives
- decomposition.

The formulae for the optimal sensitivity derivatives are based on the Karush-Kuhn-Tucker conditions (KKT conditions). The KKT conditions are necessary conditions for a particular value \mathbf{x}^* for the vector of design variables \mathbf{x} , to be a constrained local minimum. The KKT conditions are:

1. Feasibility

$$\begin{aligned} g(\mathbf{x}) &\leq 0 \\ h(\mathbf{x}) &= 0 \end{aligned}$$

2. Active constraints

$$\lambda_j g_j(\mathbf{x}) = 0 \quad j = 1, \dots, m$$

$$\lambda_j \geq 0$$

3. Extremum of the Lagrangian over the primal subspace

$$\partial F(\mathbf{w}, \mathbf{x}) / \partial x_i + \sum \lambda_j \partial g_j(\mathbf{x}) / \partial x_i + \sum \mu_k \partial h_k(\mathbf{x}) / \partial x_i = 0 \quad i = 1, \dots, n$$

where m is the number of inequality constraints, and n is the number of design variables.

$$F(\mathbf{w}, \mathbf{x}) = \sum \omega_i f_i(\mathbf{x})$$

The optimal sensitivity derivatives are designed to handle the following situation: consider a set of design decisions $\{D_1, D_2, \dots, D_N\}$. Each design decision can be thought of as determining some of the design variables x_i . If the elements of the vector \mathbf{x} are arranged in an appropriate sequence, \mathbf{x} can be decomposed into the vector

$$(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$$

where \mathbf{x}_1 is the sub-vector of design variables determined by D_1 , and so on. This is called a decomposition of the optimization problem. The product development decision-making process is studied through the analysis these decompositions. It is worth noting that there are two choices involved in selecting a decomposition:

- 1) how to group the design variables into design decisions, and
- 2) how to iterate to convergence.

There are various criteria that a design decomposition must meet. Perhaps the most important criterion is that a convergent sequence of design decisions must produce a design balancing the design goals and meeting the customers' requirements (constraints) with a minimum of iteration.

The design decisions are related to one another in the following way. Say there is a constraint

$$g_j(\mathbf{x}_1, \mathbf{x}_2, \dots) \leq 0$$

This constraint appears in both design decision D_1 and design decision D_2 . For example, say that D_1 is to be made before D_2 . It should be evident that an initial "guess" for the values of the variables \mathbf{x}_2 must be made before

$g_j(\mathbf{x})$ can be evaluated. It should also be clear that the values of the variables \mathbf{x}_1 , as determined by solving D_1 , affect the solution for D_2 . The design variables \mathbf{x}_1 are *parameters* for D_2 , and parameters are distinguished from *local design variables*. The \mathbf{x}_1 's are local design variables for D_1 .

The optimal solution to D_2 is found by varying the local design variables \mathbf{x}_2 . The idea of the optimal sensitivity derivatives is to evaluate the effect of changes in the parameters \mathbf{x}_1 on the optimal value of the objective function which can be achieved through optimization of the design decision D_2 . The purpose is to compute $\partial F/\partial \mathbf{x}_1$, with certain constraints placed on this derivative, namely

1) the constraints of the optimization problem remain satisfied, as \mathbf{x}_1 is varied, that is

$$\begin{aligned} g(\mathbf{x}) &\leq 0 \\ h(\mathbf{x}) &= 0 \end{aligned}$$

2) the solution remains optimal as \mathbf{x}_1 is varied.

The second restriction can be enforced by requiring that the KKT conditions remain satisfied. Of course, these two restrictions on the derivative may require adjustments in the optimal values of the design variables \mathbf{x}_2 to compensate for the changes in the parameters \mathbf{x}_1 .

To denote the optimal sensitivity derivative, it is convenient to make a distinction in notation between design variables, \mathbf{x} and parameters, \mathbf{p} . Then $df/d\mathbf{p}$ is interpreted as an optimal sensitivity derivative. The optimal sensitivity derivative is given by:

$$dF/d\mathbf{p} = \partial F/\partial \mathbf{p} + \sum \lambda_j \partial g_j/\partial \mathbf{p}.$$

This formula is of central importance in structuring sequential decision-making processes.

A design decision D_1 can be made before another design decision D_2 if the values chosen for the design attributes in D_1 do not make D_2 infeasible. For example, say x_1 is a design variable to be determined by decision D_1 and x_2 , also a design variable, is to be determined in decision D_2 . Say x_1 and x_2 are coupled by an inequality constraint g :

$$g(x_1, x_2, \dots) \leq 0.$$

In sequencing D_1 and D_2 there are three alternatives:

1) make decision D_1 before D_2 . x_1 will then be fixed by D_1 and will be a parameter for decision D_2 .

2) Make decision D_2 before D_1 . x_2 will be a parameter in D_1 .

3) Combine D_1 and D_2 into a single decision element.

Consider now the case where D_1 is sequenced before D_2 . Solution of D_1 will result in a change Δx_1 from the initial value for x_1 . The effect of this change on the inequality constraint g can be assessed with a first-order approximation:

$$\Delta g \sim (\partial g / \partial x_1) \Delta x_1 .$$

Thus if $(\partial g / \partial x_1)$ and Δx_1 are opposite in sign, Δg will be negative and g will be less critical in making decision D_2 (in comparison with the initial design). If $(\partial g / \partial x_1)$ and Δx_1 have the same sign, g will become more critical for D_2 if decision D_1 is made first.

Feasible sequences for the design decisions can be determined using the directions of proposed changes in the design variables in each decision and the signs of the partial derivatives of inequality constraints coupling two or more decisions together. The criteria are

F-1) If D_1 does not make (any of) the constraints of D_2 more critical, then D_1 can be sequenced before D_2 .

F-2) If D_2 does not make (any of) the constraints of D_1 more critical, then D_2 can be sequenced before D_1 .

If D_1 makes the constraints of D_2 more critical, and D_2 makes the constraints of D_1 more critical, then it may be necessary to combine D_1 and D_2 into a single decision element. If both F-1 and F-2 are met, D_1 and D_2 can be made concurrently.

There may be many possible decision sequences meeting these criteria. In an extremely tightly coupled problem, all of the initial design decisions may be combined into a single design decision by this procedure. All of the decision sequences meeting criteria F-1 and F-2 will lead to feasible designs. Next, additional restrictions on the possible decision sequences are considered. These restrictions can be shown to produce a sequence of decision elements leading to an optimal, and by extension, a Pareto-optimal design.

Determination of a sequence of design decisions leading to an optimal design requires an initial suboptimization pass through each of the decision elements. In this suboptimization pass, each of the decisions in which one of the objective functions for the design appears explicitly as a function of the local decision variables is solved in isolation. The results of

the suboptimization pass are then analyzed using sensitivity of optimal solutions to problem parameters. That analysis is used to establish whether an iteration of the decision-making procedure will progress toward an optimal design.

In constructing a decision sequence leading to an optimal design, there are again three alternatives: place D_1 before D_2 in the decision-making sequence, place D_2 before D_1 , or combine them. Let $f(x_1, x_2, \dots)$ be an objective function to be minimized in both D_1 and D_2 . If D_1 is made before D_2 , then x_1 appears in D_2 as a parameter. The sensitivity of the optimal solution to D_2 to the parameter x_1 is df/dx_1 . Directions of proposed changes in the design variables are known from the suboptimization pass, so

$$\Delta f \sim (df/dx_1) \Delta x_1 .$$

is determined.

Thus, if df/dx_1 and Δx_1 are opposite in sign, Δf will be negative. Then if D_1 is made before D_2 , f will not increase during the decision subsequent $\{D_1, D_2\}$. Any decision subsequent in which f will not increase can form part of an optimizing decision sequence. Optimizing decision sequences are built up from such subsequences, with one additional criterion: decision elements with $df/dx_i = 0$ must be placed after decision elements with $df/dx_i \neq 0$. The need for this criterion will emerge from consideration of convergence questions.

Global convergence of an effective procedure for solving an optimization problem using an optimizing sequence of design decisions is a consequence of the propositions 1, 2, and 3.

There may be several alternative design methodologies available to solve a given problem. Other considerations often enter into the decision sequencing problem, such as controlling costs associated with developing design definition or running product development tests. Thus, in applying the convergence results to synthesize a design methodology, it does not make sense to try and give a completely deterministic algorithm for selecting a decision sequence. Instead, a step-by-step process is given for constructing a design methodology which clearly indicates the points at which the design team can select among alternative design methodologies to meet economic, program milestone, product definition technology, or test schedule constraints. The role of optimization theory is to provide well-defined criteria that must be met by these alternative sequences and groupings of design choices.

Step 0. Initialization. Choose an initial design within the variable bounds and make an initial choice of decision elements.

Step 1. Evaluate each decision element to determine an optimum solution for that decision element (in isolation).

Step 2. Identify possible feasible decision sequences. If feasibility requires combination of decision elements, iterate with Step 1.

Step 3. Identify possible optimal decision sequences. Check convergence. If solution is converged, stop. If optimality requires combination of decision elements, iterate with Steps 1 and 2.

Convergence criterion: Both (i) design variables did not change during last solution pass and (ii) all optimal sensitivities are zero ($df/dx_i = 0$ for all parameters x_i) must be satisfied.

Step 4. Select a decision-making sequence that is both feasible and optimal. If D_i is sequenced before D_j , the number of parameters passed from D_i to D_j must equal or exceed the number of independent active constraints common to both decision elements.

Step 5. Find an optimal solution for each decision element in sequence. Update the values of all design variables and iterate from Step 2.

This procedure will converge to an optimal solution from any initial design within the variable bounds, provided that the decision-support procedures applied to solve the individual decision elements do so. The solution set for the procedure is defined by the condition that all $df/dx_i = 0$. This solution set is the set of Kuhn-Tucker-Karush points.

In addition, this approach provides a highly efficient technique for finding all of the Pareto-optimal design solutions. Pareto-optimal solutions minimize an objective

$$F = \sum \omega_i f_i, \quad \sum \omega_i = 1.$$

which is a weighted sum of multiple objectives which may correspond to conflicting requirements. To find all of the Pareto-optimal solutions, one would ordinarily have to solve an optimization problem for each set of values for the weights ω_i .

Information developed in the decision sequencing process can be used to avoid the reoptimization. To accomplish this, multiple objectives, f_i , are allocated to different decision elements. Then, at Step 3 above, the optimal sensitivity derivatives df_i/dp are available. An approximation to the Pareto-optimization problem having optimality conditions

$$dF/dp_i(\omega) = 0$$

is then defined.

These conditions are identical to the convergence criteria for the solution of the exact Pareto-optimization problem using the optimal decision-sequencing solution procedure. The exact Pareto-optimization problem is solved when the optimality conditions for the approximate problem are satisfied. These conditions may be satisfied by varying the relative prioritizations.

7.3 Organization of a Probabilistic Decision-making Strategy

The decomposition of a product development problem into decisions is distinct from the function and form decompositions. To a certain extent, function alternatives can be defined independently from customer requirements. It is through the decision decomposition that customer requirements are incorporated into the product.

The decomposition of complex function and form alternatives, linked by engineering theories and models, has been considered in earlier chapters of this report. The view of the product development process taken there was that decisions are essentially sequential. This model of the product development process as a sequence of deterministic decisions captures the fact that, for example, level I specifications are developed prior to level II specifications. This model reflects the fact that level II specifications are constrained by previous decisions. Thus, it is critical to determine all product and process attributes directly impacting customer requirements on level I.

In this section of the report, an alternative view of the decision-making process is presented. In this non-sequential view, a number of product and process decisions are made concurrently. Of course, product and process decisions can be made concurrently in the sequential approach (if they are decoupled). The important difference is that decisions which are quite closely linked are made in parallel in the concurrent engineering approach. By compressing the decision-making process, the concurrent engineering approach offers advantages such as reduced time-to-market and enhanced communication between upstream and downstream decision-makers. Concurrent engineering raises some technical problems, perhaps the most important of which is the problem of integrating tightly coupled decisions made concurrently.

In the sequential approach, product, process and support concepts are integrated using sensitivity information to group and sequence decisions. Concurrency implies that sequence is loosely structured, if not altogether absent. Thus, an alternative to the combination of sequencing and sensitivity analysis is needed to integrate the system concept using the concurrent engineering approach. The basic idea can be described as probabilistic engineering design, a term used by Siddall [SIDDALL]. Probabilistic engineering design ideas have also been articulated by Tse

[TSE & CRALLEY] and Ross [ROSS]. In the probabilistic design approach, product and process attributes are modelled as random variables. There is always some underlying variation in product characteristics, due to variation in production processes, maintenance procedures, or use. Thus, there is a probability distribution associated with each attribute of the product or process. In Bayesian decision theory, probability distributions are used to model uncertainty concerning the state of nature. As applied to the product development process, the "state of nature" is the vector of values of product and process attributes, as currently specified. Thus, although decisions concerning product and process attributes are made concurrently, precise values for these attributes can not be specified by these decisions. This is in complete contrast to the deterministic view. In the probabilistic view, the product development team does not specify the values for design attributes. Rather, the goal of the product development process is to influence the shape of the probability distributions.

7.4 Basic Concepts Underlying Taguchi and Tse Approaches to Design

The first key idea is:

Functional relationships transform the shape of (probability or fuzzy set) distributions.

Informally, a probability distribution is a function defined by an integral. Say x is the name of some quantity subject to variability or uncertainty. The technical term for such a quantity is a random variable. If we know the probability, $P(a < x < b)$, that x is between two numbers a and b , with $a < b$, for any two numbers a and b , we can define a "function" ϕ that satisfies

$$P(a < x < b) = \int_a^b \phi(x) dx.$$

We have to be somewhat careful here. In fact, we need to generalize the definition of a function somewhat (distributions) and use Lebesgue integration to handle all of the interesting cases. We will only work with distributions that are ordinary functions here.

Fuzzy set membership functions are similar in some ways to probability distributions. The interpretation is different, however. Embedded in the concept of a probability distribution is the concept that it is the limiting case of a frequency histogram when the sample size approaches infinity. Fuzzy distributions are free of this connotation. A fuzzy distribution is taken as the *definition* of an attribute. An example is the attribute "small". It may be useful in design to allow "small" to take on a range of values. Not all values are "equally small", however, so there is a distribution associated with "smallness". The basic idea is that "smallness" can be quantified and is given by the value of the fuzzy set membership function. In design applications, the distinction between

fuzzy set membership and probability is a matter of interpretation. The arguments we make here are based on properties of the distribution and membership functions, and so apply to both interpretations.

It is useful to distinguish between *architectural* design choices and the selection of values for design *parameters*. A system architecture is the decomposition of a system into elements, and the specification of the interfaces between those elements. Thus, architectural design choices involve the selection of system elements. For example, carburetion vs. fuel injection, diesel power vs. spark detonation, hydraulic or electromagnetic vs. mechanical power transmission, air or water cooling, DC as opposed to AC electrical systems. Even if the basic system elements are the same, changes in the interfaces may result in significant differences in architecture: front-wheel or rear-wheel drive; gull-wing, sliding, or conventional door hinges, crash protection through energy absorbing or load-transmitting structures. Mistree [MISTREE, SMITH, BRAS, ALLEN, AND MUSTER] makes a similar distinction between *selection* and *compromise* problems in design.

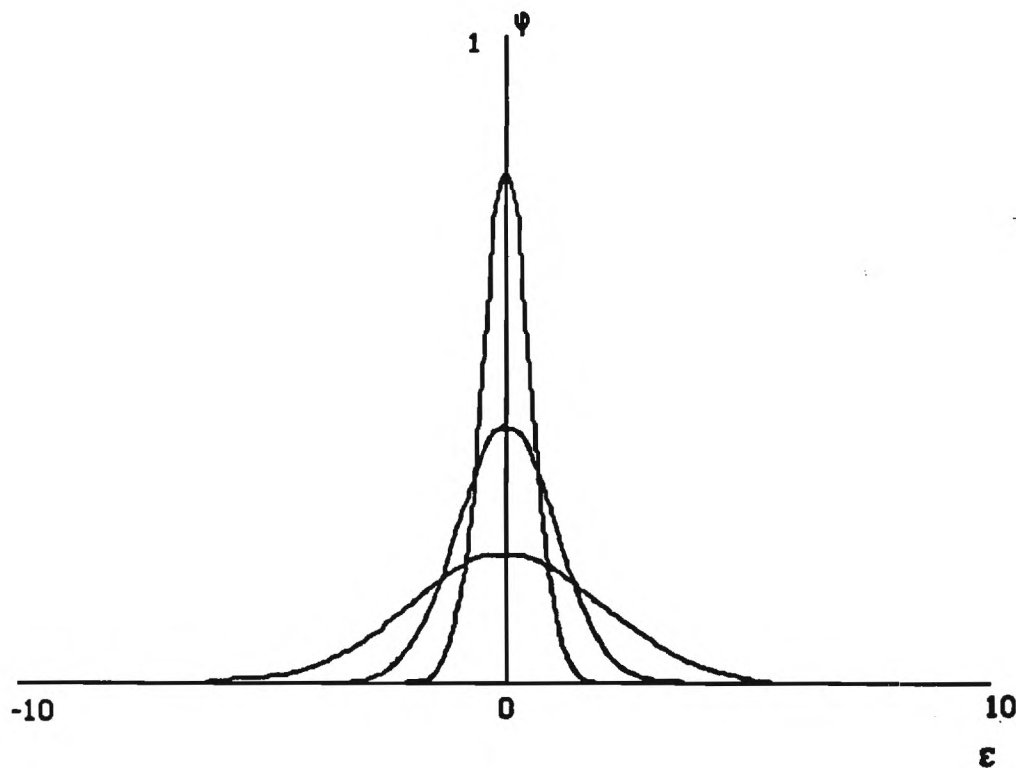


Figure 7.5. Normal Distribution of e , the Uncertainty Associated with x .

The second key point is:

Architectural design choices effect the functional relationships between design attributes.

In fact, the shape of functional relationships can also be influenced by the choice of a value for a design parameter.

For example, if a variable x is normally distributed (Figure 7.5), the variable y , functionally related to x by $y = f(x) = (x - g)^2$ has one of the distributions shown in Figure 7.6. The shape of the distribution depends on the mean value of the distribution $\phi(x)$, the variance of $\phi(x)$, and the constant g . Two curves are shown in Figure 7.6, illustrating the effect of the constant g on the transformed distribution. One distribution is highly concentrated near the origin, while the other distribution is much more spread out.

Continuing this example, let the choice of the constant g correspond to a design decision. Also, let y be a product (or process) attribute subject to uncertainty (i.e. to be determined downstream), and further let x be a product characteristic, perhaps a measure of merit in meeting a requirement, which must be met with a prespecified variance. Clearly, the choice of the constant, g , resulting in the "peaky" distribution of Figure 7.6 has severely restricted the possible choices for y . If y is a parameter characterizing a manufacturing process, the value of g leading to the "flat" curve is better from a producibility standpoint. Why? The manufacturing process planner has a wider range of options. The shape of the distribution of y associated with the required distribution of the controlled product characteristic, x , is flatter.

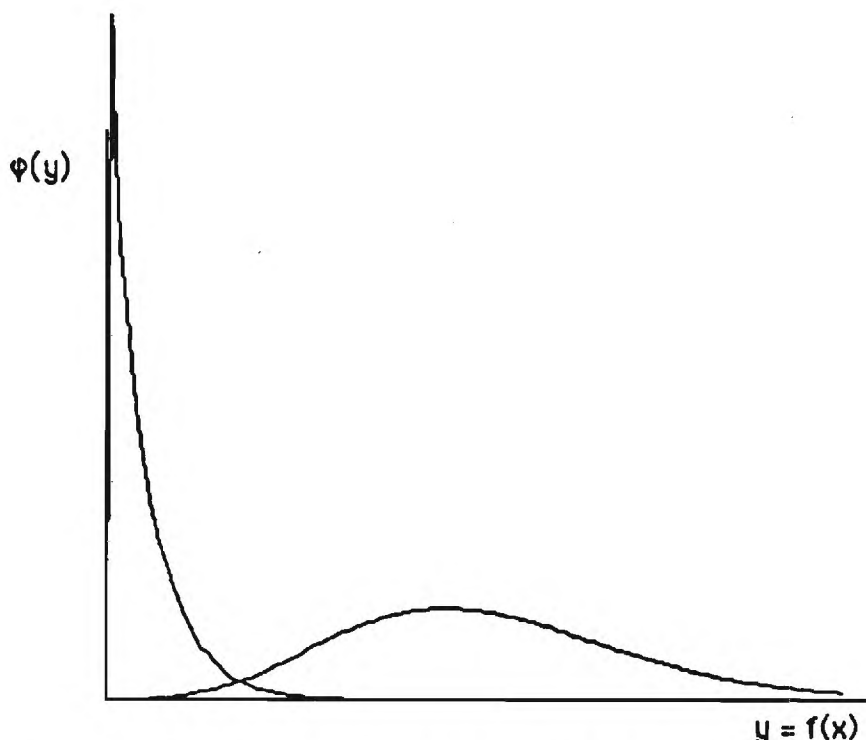


Figure 7.6. Functions Transform the Shape of Probability Distributions.

This idea has important consequences for design. Before turning to these, we note that the effect of functional relationships on probability distributions is a consequence of a well-known result in probability theory [SCHMETTERER].

Consideration of these two points leads to a new view of system design:

System design is the shaping of probability (or fuzzy set) distributions.

This point of view is clearly evident in Tse's work [TSE & CRALLEY]. It is less readily apparent in the "classical" Taguchi method [ROSS, but is also present there. The basic idea of Taguchi's method is to shape the distribution of a variable that is a function of the product characteristics, the *loss function*. In the classical Taguchi approach, this distribution is characterized by its location (mean) and scale (variance). Location and scale are clearly shape descriptive parameters. The orthogonal arrays for noise factors are used to estimate the variance in the loss function associated with noise.

Deterministic relationships, such as the relationship between required and available power, couple attributes of the design. Over time (within a single product instance), and between product instances, vehicle attributes such as [total efficiency], [drag coefficient] and [vehicle weight] have some uncertainty associated with them. In addition, there is uncertainty associated with incomplete product specifications in early product development decisions.

$$\begin{aligned}
 & \frac{[\text{motor torque}] [\text{transmission ratio}]}{[\text{dynamic tire radius}]} [\text{total efficiency}] = \\
 & [\text{vehicle weight}] \left[[\text{coeff. of rolling resistance}] \cos[\text{incline angle}] \right. \\
 & \quad + \sin[\text{incline angle}] \\
 & \quad \left. + [\text{rotational inertia coeff.}] \frac{[\text{acceleration}]}{[g]} \right] \\
 & + 0.5 [\text{air density}] [\text{speed}]^2 [\text{drag coefficient}] [\text{vehicle cross-section area}]
 \end{aligned}$$

For the moment, say one of the vehicle attributes, such as [motor torque], is selected as a dependent variable. Then, the required/available power relationship transforms the uncertainties in efficiency, drag coefficient, and weight into a distribution on the dependent variable, torque.

More complex situations are typical. For example, many technical relationships of importance in product development are themselves subject to some uncertainty, for example, the relationship between crash structural integrity and curb weight (based on data for 1988 cars), Figure 7.7, or the relationship between curb weight and fuel efficiency, Figure 7.8.

For a given value of the structural integrity rating, a range of variability in vehicle curb weights is seen. There is also a range of structural integrity ratings associated with a given curb weight. It seems somewhat simpler to conceptualize the distribution of curb weight (a continuous random variable) as a function of structural integrity rating (a discrete variable).

The non-deterministic relation between EPA mpg and curb weight is somewhat simpler to conceptualize, since both variables are continuous.

Decisions are refined by acquiring additional information about the life cycle concept for the motor vehicle. In a convergent product development process, the variance of the distributions associated with product attributes decreases as the concept is refined. The distribution on a product attribute may be refined by acquiring additional data. Refinement of the distribution on a closely coupled attribute may also allow the distribution to be refined. This process is illustrated in Figure 7.9.

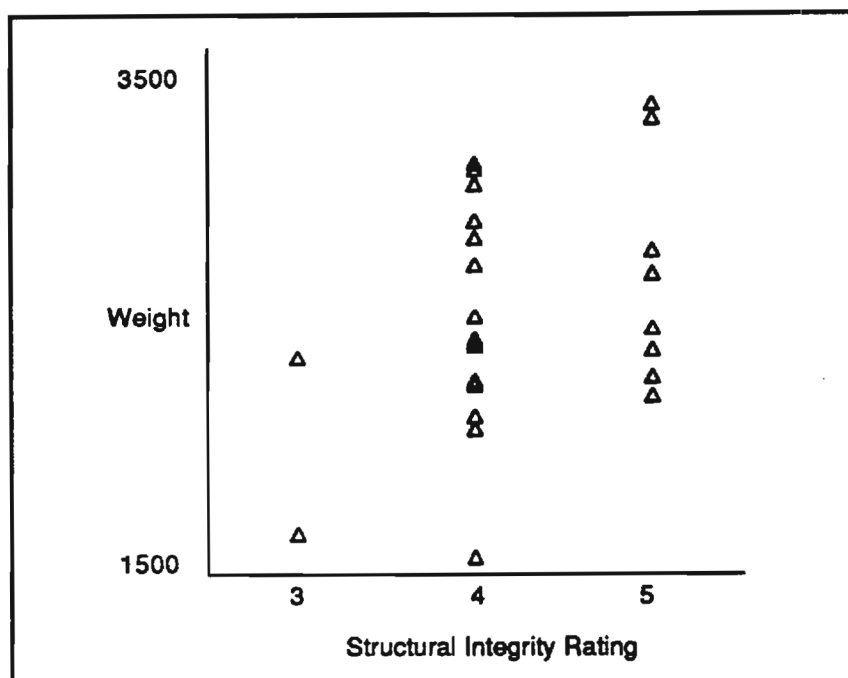


Figure 7.7. Non-deterministic Relation of Curb Weight to Crash Test Structural Integrity Rating.

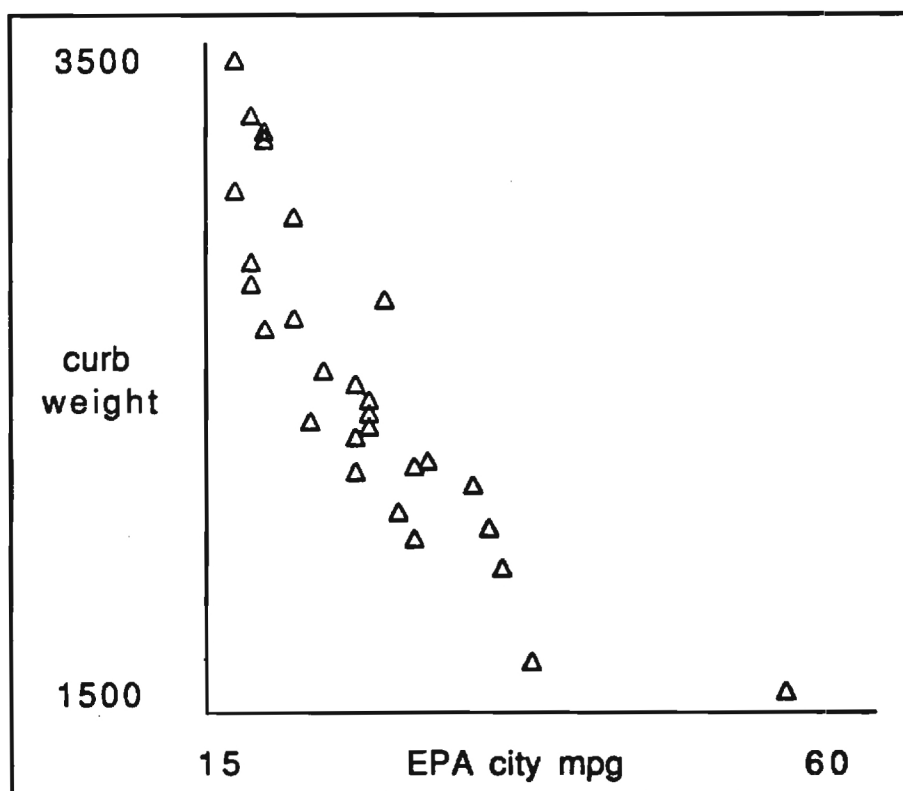


Figure 7.8. Non-deterministic Relation between EPA mpg and Curb Weight.

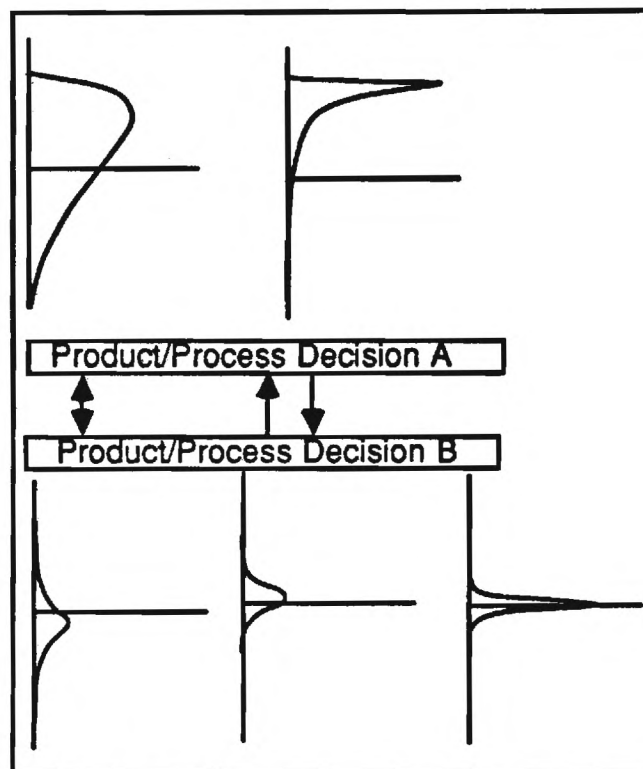


Figure 7.9. Decision Refinement Reduces the Variance of Distributions on Product Attributes under Uncertainty.

7.5 References

- J.L. Cohon (1978). *Multiobjective Programming and Planning*, Academic Press, New York.
- M.A. Kolb (1990). *An Investigation of Constraint-Based Component-Modeling for Knowledge Representation in Computer-Aided Conceptual Design*, Ph. D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA.
- Y. Leung (1988). *Spatial Analysis and Planning under Imprecision*. North-Holland/Elsevier, Amsterdam/New York/Oxford/Tokyo.
- F. Mistree, W.F. Smith, B. Bras, J.K. Allen, and D. Muster (1990). *Decision-Based Design: A Contemporary Paradigm for Ship Design*, to be presented at the Annual Meeting of the Society of Naval Architects and Marine Engineers, San Francisco, CA.
- J.E. Rogan and W.E. Cralley (1990). *Meta-Design: An Approach to the Development of Design Methodologies*, IDA Paper P-2152, Institute for Defense Analyses, Alexandria, VA.

P. J. Ross (1988). *Taguchi Techniques for Quality Engineering*. McGraw-Hill Book Company, New York, et al.

L. Schmetterer (1974). *Introduction to Mathematical Statistics*. Springer-Verlag, New York/Heidelberg/Berlin.

J.N. Siddall (1983). *Probabilistic Engineering Design: Principles and Applications*. Marcel Dekker, New York/Basel.

J. Sobieszczanski-Sobieski, J.-F. M. Barthelemy, and K.M. Riley (1982). *Sensitivity of Optimum Solutions to Problem Parameters*, AIAA 81-0548R, AIAA Journal, v. 20, n. 9, pgs. 1291-1299.

D.V. Steward (1981). *The Design Structure System: A Method for Managing the Design of Complex Systems*, IEEE Trans. on Eng. Mgmt., v. EM-28, n. 3.

E. Tse and W.E. Cralley (1989). *Management of Risk and Uncertainty in Product Development Processes*, IDA Paper P-2153, Institute for Defense Analyses, Alexandria, VA.

7.A.1 Solution of Requirements Negotiation Problem using Methodology F.

The formulation of subproblem F_3 in terms of f_1^{opt} and f_2^{opt} suggests the use of optimal sensitivity derivatives. Thus, in applying methodology F to the requirements negotiation problem, the first step is to solve subproblems F_1 and F_2 , estimating the optimal sensitivity derivatives of f_1^{opt} and f_2^{opt} with respect to l , w , and h from the solutions to these subproblems. To do this, values for the Lagrange multipliers for the constraints in which l , w , and h appear explicitly are needed.

Solving F_1 to determine a value for the Lagrange multiplier of the constraint

$$g_1 = c_1 - lwh,$$

note that an optimal solution to this problem must satisfy the third KKT condition:

$$\partial f_1 / \partial c_1 + \lambda_1 \partial g / \partial c_1 = 2(c_1/10 - 1)(1/10) + \lambda_1 = 0.$$

Then

$$\lambda_1 = (1/5)(1 - c_1/10).$$

where λ_1 is the desired Lagrange multiplier.

Determining a value for the Lagrange multiplier of the constraint

$$g_2 = 2(lw + wh + lh) - c_2$$

appearing in subproblem F_2 , the third KKT condition is applied once again

$$\partial f_2 / \partial c_2 + \lambda_2 \partial g / \partial c_2 = (1/3)(c_2/6 - 1) - \lambda_2 = 0.$$

Thus,

$$\lambda_2 = (1/3)(c_2/6 - 1).$$

The information required to solve subproblem F_3 is now in hand. The optimal solution to problem 2(a) is a function of l , w , and h . Denote this function by $f_1^{\text{opt}}(l, w, h)$. Compute the optimal sensitivity derivatives $D_l f_1^{\text{opt}}$, $D_w f_1^{\text{opt}}$, and $D_h f_1^{\text{opt}}$. (The optimal sensitivity derivatives can be computed from partial derivatives of the objective function and constraints with respect to the decision variables at the optimal solution.) To emphasize that l , w , and h are parameters for subproblems F_1 and F_2 , denote them by

$$l = p_1, w = p_2, \text{ and } h = p_3.$$

Use the optimal sensitivity derivatives to construct an approximation

$$f_1^{\text{opt}}(p_1, p_2, p_3) \sim f_1^{\text{opt}}(p_1^0, p_2^0, p_3^0) + \sum D_{p_i} f_1^{\text{opt}} \Delta p_i$$

about the point (p_1^0, p_2^0, p_3^0) .

Define $f_2^{\text{opt}}(p_1, p_2, p_3)$ in the same way, and approximate

$$f_2^{\text{opt}}(p_1, p_2, p_3) \sim f_2^{\text{opt}}(p_1^0, p_2^0, p_3^0) + \sum D_{p_i} f_2^{\text{opt}} \Delta p_i.$$

To compute the optimal sensitivity derivatives, the reasoning of [SOBIESKI, BARTHELEMY & RILEY] is followed:

$$df/dp = \partial f / \partial p + \sum \partial f / \partial x_i \partial x_i / \partial p$$

The Kuhn-Tucker-Karush optimality conditions deliver

$$\partial f / \partial x_i + \sum \lambda_j \partial g_j / \partial x_i = 0$$

substituting this into the expression for the total derivative of f with respect to p ,

$$df/dp = \partial f / \partial p - \sum_i \sum_j \lambda_j \partial g_j / \partial x_i \partial x_i / \partial p$$

Requiring that the constraints remain satisfied as p is varied gives

$$dg_j/dp = \partial g_j / \partial p + \sum \partial g_j / \partial x_i \partial x_i / \partial p = 0$$

This permits one more simplification to df/dp ,

$$df/dp = \partial f / \partial p + \sum_j \lambda_j \partial g_j / \partial p$$

Applying this method to differentiate the optimal value of f_1 as the parameter p_1 is varied,

$$D_{p_1} f_1^{\text{opt}} = \partial f_1 / \partial p_1 + \sum_j \lambda_j \partial g_j / \partial p_1 = l(-wh) = (1/5)(1 - c_1/10)(-wh)$$

where λ is the Lagrange multiplier of the constraint $g = c_1 - lwh \leq 0$ in problem 2(a).

In a similar computation, determine

$$D_{p_2} f_1^{\text{opt}} = (1/5)(1 - c_1/10)(-lh)$$

and

$$D_{p_3} f_1^{\text{opt}} = (1/5)(1 - c_1/10)(-lw)$$

Then

$$f_1^{\text{opt}}(p_1, p_2, p_3) \\ - f_1^{\text{opt}}(p_1^0, p_2^0, p_3^0) + (1/5)(1 - c_1/10)(-wh)\Delta p_1 + (1/5)(1 - c_1/10)(-lh)\Delta p_2 \\ + (1/5)(1 - c_1/10)(-lw)\Delta p_3.$$

Similar computations give

$$D_{p_1} f_2^{\text{opt}} = (1/3)(c_2/6 - 1)(w+h)$$

(the other derivatives are computed in exactly the same way), and

$$f_2^{\text{opt}}(h_0, c_1) \\ - f_2^{\text{opt}}(p_1^0, p_2^0, p_3^0) + (1/3)(c_2/6 - 1)(w+h)\Delta p_1 + (1/3)(c_2/6 - 1)(l+h)\Delta p_2 \\ + (1/3)(c_2/6 - 1)(l + w)\Delta p_3$$

Using the approximations to $f_1^{\text{opt}}(p_1, p_2, p_3)$ and $f_2^{\text{opt}}(p_1, p_2, p_3)$, determine capacity = $c_1(w_1, w_2, p_1, p_2, p_3)$ and cost = $c_2(w_1, w_2, p_1, p_2, p_3)$ as solutions to the minimization problem posed for subproblem F_3 , restated here with appropriate variables.

minimize:

$$F = \omega_1 f_1^{\text{opt}}(p_1, p_2, p_3) + \omega_2 f_2^{\text{opt}}(p_1, p_2, p_3)$$

subject to:

$$p_3 - 2 \leq 0$$

$$\omega_1 + \omega_2 = 1$$

$$\omega_1, \omega_2 \geq 0$$

where the design variables are now:

$$l = p_1, w = p_2, \text{ and } h = p_3.$$

Chapter 8 MULTIVIEW DESIGN APPLICATION

8.1 Introduction

This chapter summarizes the efforts accomplished in the selection of an Object Oriented Programming paradigm to enable the testing of ideas and concepts generated during the decomposition of the vehicle into form and function hierarchies.

8.4 Applications

8.4.1 Form-Function Test with Smalltalk

An initial test project developed in the Smalltalk object oriented environment investigated a method to link abstract functions with design variables in an automobile example. An adjustable parametric equation approach was selected to model the system. The test project was a very simple two dimensional model of a vehicle that included only six design parameters. The six parameters were the length and width of the cargo, passenger, and engine compartments in a vehicle. The reason these three compartments were selected is because it was felt that the overall shape of the vehicle was a direct function of these three compartments. The system consisted of three dials that measured abstract functions, namely our three base level functions: Please, Transport, and Protect. The range on the dials was from 0 to 100%, and by pointing with a mouse and varying one or more of the dials, the overall shape of the vehicle changed. Figure 8.1 illustrates the computer screen with the initial vehicle and the dials at nominal 50% values. Note that the vehicle lines were approximated using bezier splines and several control points. The original model consisted of three rectangles located next to each other and representing the three compartments. The design variables were the length and height of each rectangle representing a compartment.

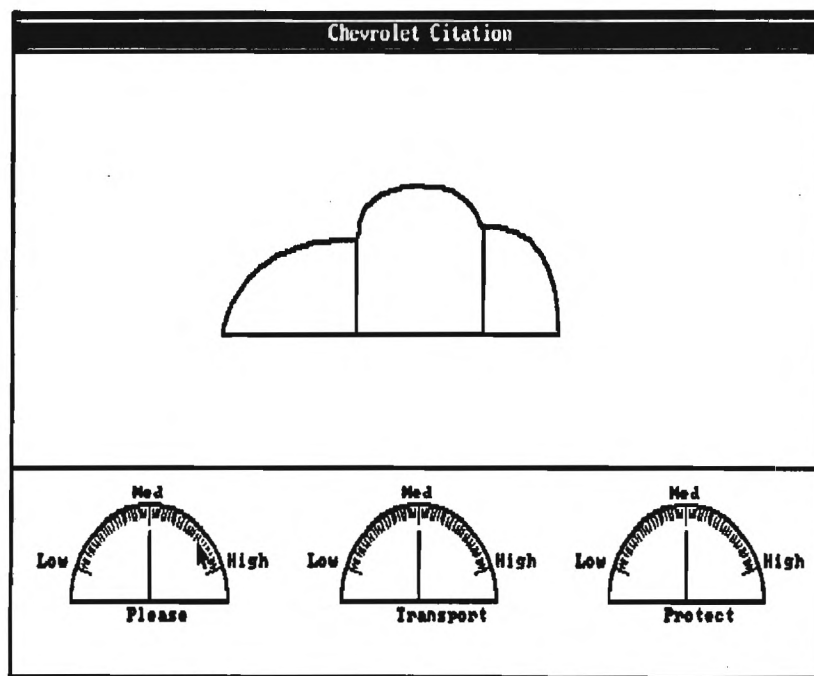


Figure 8.1. Smalltalk environment and test vehicle at 50% of functional values.

Parametric relationships were chosen to model the form of the design. Parabolic equations were used to model the system. The equations were broken down by both function and design variable. One parabolic equation was developed for each unique pairing of function and design variable. The current model used eighteen relationships or parabolic equations. For each parabolic equation, three points were used. For a real design situation, hundreds of points may be used. The middle point always represented 1 or the nominal value for the design variable. The other two points represented the end points of the design variable relative to the upper and lower bound of the function considered. Figure 8.2 illustrates the same vehicle with an increase in the level of protection which results in a longer chassis, larger engine and cargo compartments.

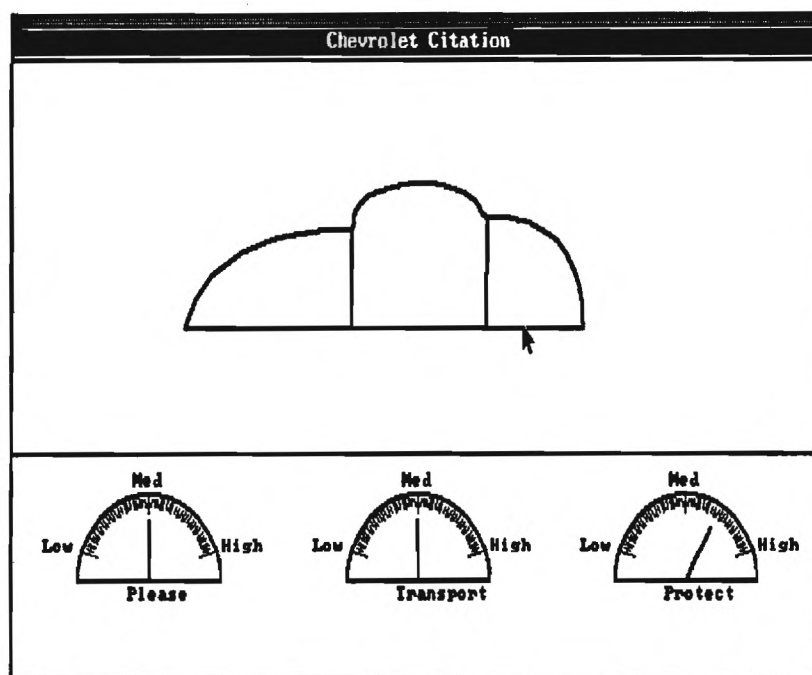


Figure 8.2. Smalltalk environment and test vehicle at 70% of Protection value.

As mentioned above, in the current simplified model, three functions and six design variables were considered so that eighteen parametric equations were created. The equations were built arbitrarily from the developers' impressions of how the form varied with the function. An extensive development test would require more careful study, knowledge based upon experts, and many consumer impressions. The cumulative change of each design variable was combined by multiplying the parametric contribution for each of the functions considered by the nominal value for each of the design variables.

$$DV1 = \text{Please}(DV1, \%) * \text{Transport}(DV1, \%) * \text{Protect}(DV1, \%) * (\text{Nominal } DV1 \text{ Value})$$

This allows one to always get back to the nominal design by setting the function dials to 50%. If two dials are set to 50% and the other dial is

allowed to vary, the model will change only with respect to the one varying function only.

An additive method of summing up or combining functions was not considered, but it could be considered in the future. The nominal value of a function does not need to be at 50%; 50% was convenient for the simplified model.

Another parametric scheme that was considered was to use neural nets to define the relations. An optimal result may not be produced, but neural nets would help to manage the complex parametric relationships needed to model the transformation. The method considered is an extremely simple model of the parametric readjustment that occurs in a neural network.

8.4.1.1 Future Considerations

The simple model proved the feasibility of a function to form relationship, and was used as a basis for working with objects in the design decomposition. We also investigated how functional parameters should be set. For instance, the vehicle, according to the decompositions outlined in the previous chapters, accomplishes three primary functions: Please, Transport, Protect. How can we quantify these functions? Is the sum of the functions equal to 100%, or are they independent? We are still trying alternatives to access the results. An interesting approach is for instance to set a total of 100%. Then, a sports car could see its baseline functional breakdown at Transport: 30%, Please: 60% and Protect:10%. This characterizes a car that is more designed for pleasure than transport. But, how do these values translate to lower levels in the form hierarchy? Let's look at the second level form decomposition, we have the transportation/support systems: Chassis and structure; Entertainment/driver interface systems: exterior and interior systems; Power/transformation systems: Engine, transmission, and finally Computer control systems. Looking at the levels of Transport, Protect, and Please, the chassis could be broken down for instance in 20% Transport, 75% protect and 5% Please. Interior systems would be broken down into 0% transport, 80% please and 20% protect. Breaking down the chassis further into a third level form, a side support beam would be broken down into 90% protect and 10% transport. The challenge now is to modify the transport function at the base level, and see an increase in the size of the side support beam of the chassis. Then, could we go back up the tree and decide that a size change in this specific beam is going to affect the transport function by such a value? These are questions that are still under investigation, and that will be possibly addressed in further research.

Additionally, considering the above model, other parametric relationships should be considered, a more extensive example should be

conducted and possibly, an investigation of relationships modelling using connectionist systems should be considered.

Having completed the proof of concept, other techniques of Object Oriented representation needed to be investigated to address decomposition techniques, and data storage.

8.4.2 Work Related to OOPM

Extensive research was conducted before building an OOPM. There are a surprising number of good examples to learn from [13]. Some of the work had merit in the theory of representing the design; whereas, other works were classic examples of proper object-oriented programming practice. Some systems are commercial applications and others are proof of concept systems.

Lately, there have been many initiatives to represent engineering design and systems in terms of discrete objects. Some projects have been application domain specific, others have concentrated on the physical information only, and others have concentrated on the analysis of information only. Many disciplines converge simultaneously when trying to operate on the problem of design representation.

The Sketchpad project done at MIT in the 1950s has been considered as the beginning of the modern CAD system. Sketchpad used constraint propagation and a primitive form of constraint modelling. There is no object-oriented programming in the Sketchpad project.

The next major contribution to design decomposition using objects and constraint representation used the Smalltalk OOP system. The Thinglab project was conducted by Alan Borning [3] of Stanford University in 1979. Thinglab was a system which allowed a user to create new classes of objects along with the constraints associated with the object. The Thinglab project did not center on how to decompose an engineering design but was primarily concerned with creating a general purpose system to define classes of things and the constraints which define the behavior of these "things". Thinglab represented constraints or association relationships using a constraints class as one special class of its own, where specific information was added to make the constraint specific to the object being defined.

In 1989, an object-oriented programming project called "Rubber Airplane" [15] was completed at MIT. This project applied some of the concepts of Sketchpad and a few of the concepts of Thinglab and constraint management to an Airplane example. The "Rubber Airplane" allowed a fairly flexible relationship object; the relationship object defined the behavior of the interface of two physical objects as the model stretched and contracted. This required a lot of specialized programming for the interface

of each of the two parts. The other drawback of the detailed relationship objects is that if one wanted to try new configurations for the parts and sub-assemblies, new special programming was required to build the new configuration. The relationship of the constituent parts was not as flexible as the Thinglab project, but the "Rubber Airplane" was a more realistic tool for engineering design.

Yoshikawa, from the University of Tokyo, took a much more theoretical approach with the General Design Theory [32]. The General Design Theory introduces the intersection of function spaces. The unique intersection of function spaces define different products in design. The General Design Theory is intuitively satisfying, but it lacks a firm theoretical basis, or extensive experimental justification. Later, Yoshikawa enhanced the General Design Theory by applying it to the Metamodel [29] project. The Metamodel is an attempt to build an analysis independent representation of a design. The Metamodel generates new views and interfaces to the views by instantiating a view from the combination of the analyses to be conducted, the class of the device, and the processes that the device must be exposed to.

Nam Suh [26] from MIT expanded on the form-function transformation concepts that were being developed by Yoshikawa. Suh hypothesized that there is a transformation between function and form and between form and fabrication. A proper transformation could ultimately allow the design to progress directly from function to fabrication. Dr. Suh applied the theory to simple examples in material science. Others which have developed and advanced this theory are Rinderle [21] from Carnegie Mellon University and Pun/Colton [20] from the Georgia Institute of Technology.

Today, some commercial applications contain some of the theories set forth in the earlier works. Some parametric modelers such as Parametric Technologies' PRO/Engineer [19] and The SDRC Geomod Solid Modeler [25] use the technologies of form decomposition through features and constraint modelling to link the features of the forms. ICAD [12] has similar features imbedded in a parametric solid modeler; it adds more powerful knowledge based capabilities, but it is more tedious to use than a solid modeler.

These approaches were studied, and an Object Oriented Programming Methodology (OOPM) was derived.

8.4.3 Building the OOPM

The OOPM was built to model the decomposition needed in engineering design. The OOPM is based upon the concepts outlined above. The OOPM is an engineering analysis representation model developed on

Smalltalk. The geometry and surface information are, for the moment, assumed to be external to the system.

The goal of the model was to represent and decompose the engineering design. The design is broken into Parts and Assembly classes (Refer to Figure 8.3). Each Part and Assembly has a PartTransformation class. A part transformation can be thought of as a transformation matrix as explained by Suh. The part object's part transformation can be thought of as an elemental part transformation. The assembly part transformation can be thought of as a transformation tying the part and sub-assemblies together. The assembly transformation includes the interactions of the lower level parts including impedance functions of the Energy, Material and Information interactions.

The PartTransformation class is divided into three main divisions. The Relation class is the grouping of objects which relate the features of the part or assembly being modelled. The relations would come from analyses, expert system rules, equations, and rules of thumb. The Yoshikawa Meta-Model would instantiate relations depending on the class of the part and the modelling analysis requested. The second class of objects is the Feature class. The features are the minimum attributes that are needed to describe the part or assembly. The features are assumed to come from a feature-based modeler. The features can be thought of as the dials that one would adjust the design to achieve the desired effect.

The last class of objects is the StateScenario class. The StateScenario is analogous to the load case in Finite Element Analysis. The StateScenario allows one to simulate and document the operation and test cases for the design. The StateScenario is further broken down. In design, it is often the case that one partially knows the input and output of the system. The goal of the design is to refine the inputs and output effects by adjusting or tweaking various parameters; that is the basis for the design of the StateScenario. The InputVector includes the input effects. The OutputVector includes the output effects. The StateVector represents the state of the system during the current simulation. The StateVector typically contains the parameters that one adjusts to get the favorable transformation.

Other important classes are the Attributes class and the Dependency classes. The Attribute class is a catalog of design variables and behavior variables that can be used to describe Dependency and Relation objects. The Dependency class is the catalog of the variables used in StateScenarios and Features in the design process. The Dependency objects are variables that include an Attribute object, but also have more semantic meaning and numeric values for the problem at hand.

ONT/Smalltalk Conventions



111

8.4.4 Multiple Views

The next major section of the data representation of the OOPM is the View class. The View class stores the **alternative views of the design**. The class of views is contained inside the Assembly class. The view represents the alternative ways of viewing the design from the current, selected assembly down to the lowest level in that branch of the system. The View class is comprised of the CircuitBlock, the Link, and Interface classes. The CircuitBlock class represents the new self-contained sub-problems. The Links class is a quick reference to view the hierarchical nature of the CircuitBlocks for that view. The Interface class contains the interfaces of Dependency objects from one circuit block to the other circuit block. The Interface portrays the critical interactions of the circuit blocks. The views are still under development in the OOPM, and no example is available at this time.

8.4.5 Example

A simple design case was implemented in the OOPM for testing purposes. It is expected to use the data derived from the decompositions above to more thoroughly test the OOPM. This example illustrates some of the capabilities of the system, and some of the uses that were expected from it.

Figure 8.4 illustrates a Smalltalk screen with a subset of the Engine Assembly coded in the OOPM. The screen on the top left lists the assemblies. Note that each assembly is automatically assigned an ID number which will be later used for the different views and the decompositions. The top center screen lists some of the parts associated with the cylinder block which is highlighted. The bottom left screen is an output screen that display some of the attributes of the assemblies and a description field. The relation screen lists the relations that are associated with the assembly. Note also that the ASSEMBLY field is highlighted, which means that the object considered is an assembly, not a part.

General Attributes and Dependencies		
19 64 bit Crank Angle		
OOPM Viewer for Product: Test Engine		
9 Total Assembly 18 Basic Engine 15 Cylinder Block 18 Lubrication System 16 Piston 12 Ignition System 14 Exhaust System 13 Fuel System 11 Cooling System	5 Upper Head 9 Oil Drain Plug 8 Oil Pan Bolt 1 4 Main Block 7 Oil Pan 6 Head Bolt	
		TRANSFORMATION
		PART ASSY
Assembly ID: 15 Assembly NAME: Cylinder Block Version: A DESCRIPTION: The major housing of the basic engine		Relation ID: 134 Relation NAME: Compression Ratio Version: Weight: 2

Figure 8.4. Test engine screen in Smalltalk with Assembly selected.

Should one click on the PART field, the information about the oil pan bolt 1 part is displayed in the center right field. (Figure 8.5)

General Attributes and Dependencies		
19 64 bit Crank Angle		
OOPM Viewer for Product: Test Engine		
9 Total Assembly 18 Basic Engine 15 Cylinder Block 18 Lubrication System 16 Piston 12 Ignition System 14 Exhaust System 13 Fuel System 11 Cooling System	5 Upper Head 9 Oil Drain Plug 8 Oil Pan Bolt 1 4 Main Block 7 Oil Pan 6 Head Bolt	
		TRANSFORMATION
		PART ASSY
Assembly ID: 15 Assembly NAME: Cylinder Block Version: A DESCRIPTION: The major housing of the basic engine		Part ID: 8 Part NAME: Oil Pan Bolt 1 Version: DESCRIPTION:

Figure 8.5. Test engine screen in Smalltalk with Part selected.

The system is still under development. It was designed to allow easy input of hierarchical information derived in the form and function decompositions, and was expanded upon to identify holes and incomplete information. For this purpose, an additional feature was included in the system: the ability to superimpose decomposition methods and reorganize the data.

8.4.6 Decomposition

In searching for decomposition techniques, many researchers have domain specific impressions of decomposition. Our study attempted to seek out the application independent aspects of decomposition.

Every system is a unique connection of its parts and their interactions. A more complex system may have sub-parts and interconnections on many levels. This is a very simple model of a system, but it yields useful results. The system was modelled using Graph theory which can be attributed to Euler (circa 1736).

Using Graph theory, the system is composed of vertices and directed arcs. A vertex is some sort of design variable, evaluation, enabling event, or thing that must be evaluated. The arcs link the vertices together. One arc can link two vertices at most. The arcs are a result of some sort of relationship between vertices. An arc may connect vertices due to an engineering analysis, a corporate policy, an equation, or an expert system rule. The reason why a relation exists is called the semantics of the relationship. As a side effect, a structure of arcs are built that can be viewed independently of the semantics of the problem. In modelling a complex system, a complex graph of vertices and directed arcs could be built. The problem of decomposition is to control this topology of arcs to the advantage of the users of the system.

Now, what are the tools available? Given the assumptions about the modelling of a complex system, there are only a few major strategies that can be invoked. One could rebuild the vertices or enabling conditions so that the interconnections are simpler or less costly; we will call this **Vertex Decomposition**. This is analogous to the to the work being done with parametric feature based modelers.

Next, one could try to manipulate the topology of the structure of the problem to minimize the complexity and interconnections of the problem; this will be called **Structural Decomposition**. Structural decomposition is not very quantitative; the Steward [24] method of decomposition is a simple structural decomposition technique to implement.

Lastly, one could factor the semantics of the problem into account to reduce the redundancies of the problem; this method will be called

Semantic decomposition. The **Semantic decomposition** can be further broken into two methods. The first Semantic decomposition methods use numerical methods which require information about the numerical information of the relationship. The numeric information is manipulated without truly understanding the semantics; examples of a numeric Semantic Decomposition methods would be the Dantzig-Wolfe decomposition method and General Aggregation Decomposition methods. The qualitative Semantic decomposition method would require one to know or be cognizant of the semantics. New relationships or redundant relationships could be manipulated to simplify the topology of the design process or to remove vertices that may be unneeded or redundant. Examples of qualitative Semantic methods would be algebraic evaluation and the Kron Tearing Analysis.

The OOPM has been interfaced with the Steward's Structural decomposition which Rogers [22], from the NASA Langley Research Center, had implemented. The Relations class was implemented with a tag to identify the relations which effect a certain discipline, i.e., structural analysis, packaging, marketing. A new view is generated depending on the discipline. The Views have been designed so that the PartTransformation class and the View class operate on the same relations and dependencies to reduce the ambiguity of the final design.

An example of the decomposition algorithm is illustrated in the next two figures: Figure 8.8 and 8.9. Initially, the relations are very random. Figure 8.8 illustrates the linkages between functions and design variables. The numbers on the both sides represent a relation number. For instance, relation 49 says that piston mass is a functions of variables that are also in relation 17, 26, 61 and 69. The links are in both directions, i.e., feed forward, and feed backward. The aim of the decomposition is to reorder the relations such that the feed backward links are minimized if not eliminated. Ultimately, if links backward cannot be eliminated, then these relations are tightly coupled and should be addressed together. Otherwise, relations should only feed forward. Figure 8.9 illustrates the relations after decomposition through the DEMAID system. Note that the feedback loops have been reduced to a minimum, and the order of the design in this particular example should be: 1 or 2 (at the same level, the most important relations are the materials costs and fuel consumption. The third relation is time, the fourth is effective work available. The coupled relations (39 and 40) are piston mass and pure oscillating mass.

The example was filled with arbitrary data for checking purposes. It should be tested extensively with real data from a vehicle to see the value of the decomposition technique.

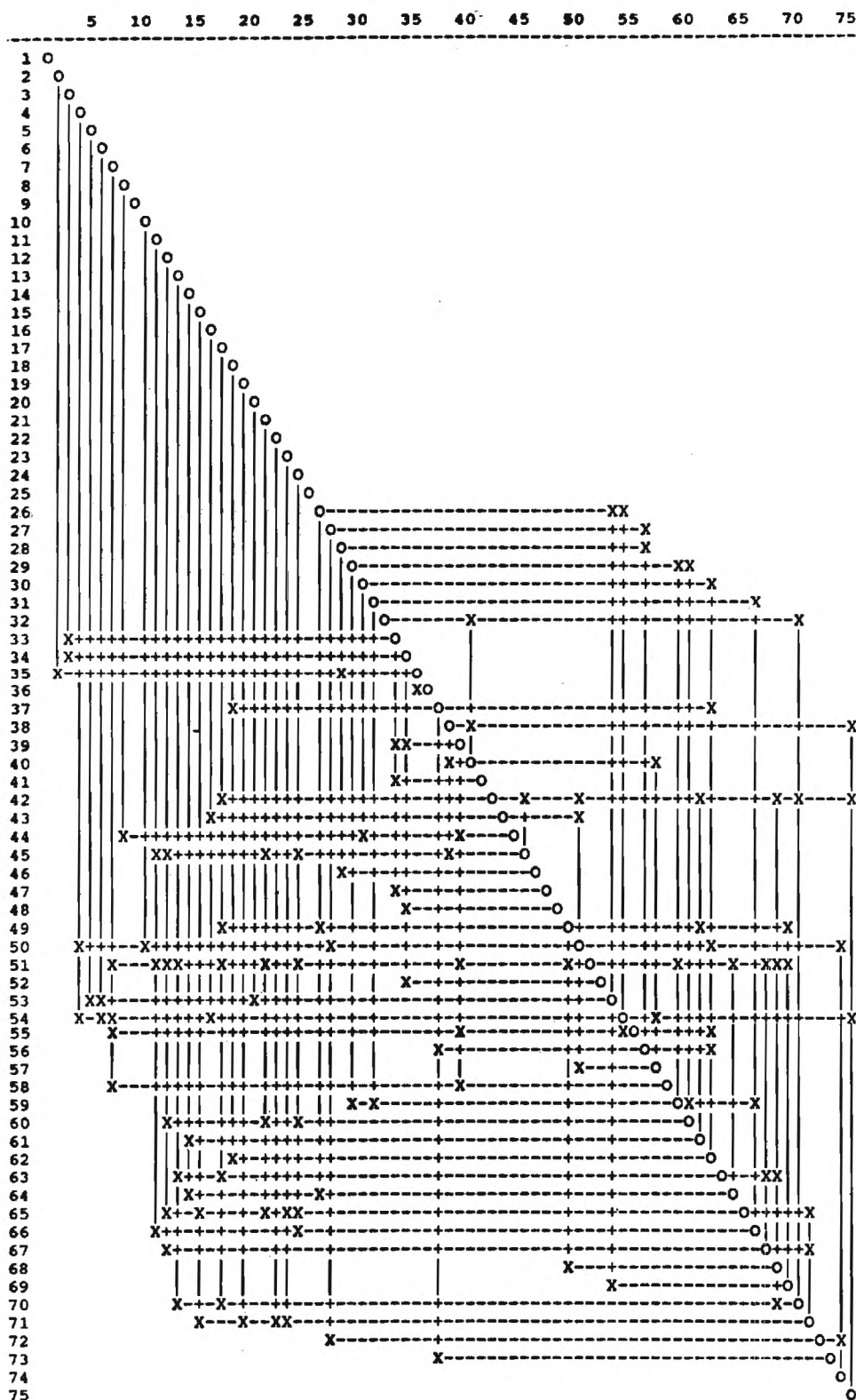


Figure 8.8. Relations diagram before the application of decomposition.

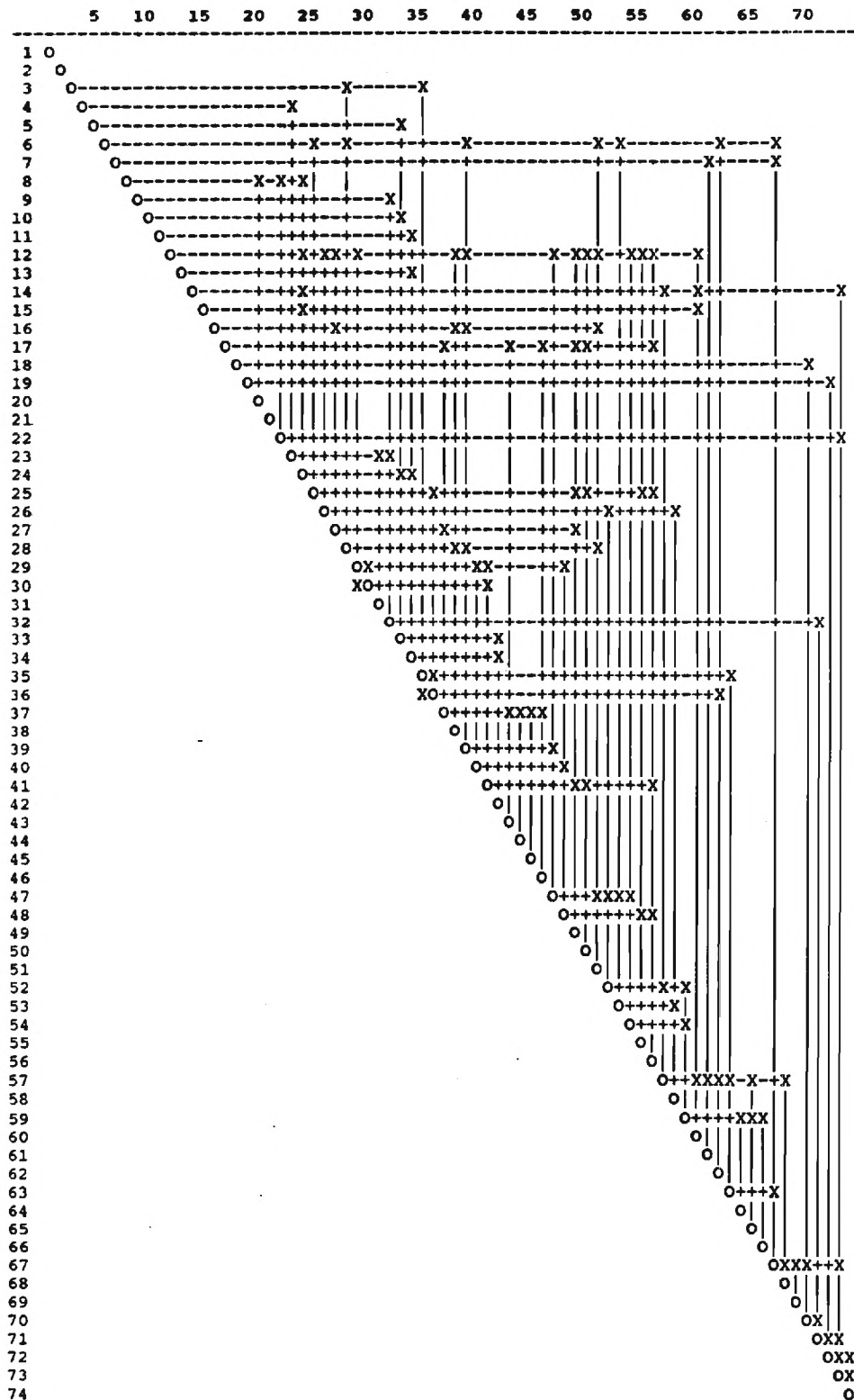


Figure 8.9. Relations diagram after decomposition through Demaid.

8.5 Summary

The present Demaid system is directly linked to the OOPM. One can enter the data in the Object Oriented Parametric Modeler and then run Demaid to reorganize the data. However, more work needs to be done to investigate the Steward Structural Decomposition. A feature based modeler should be connected to the OOPM as well as an expert system to feed relations into the system. A GenericConcept class could be developed to instantiate rules and relations to the OOPM. This is similar to some of the features in the Yoshikawa Metamodel. An interface should be developed to the Function-Form Mapper to obtain the output forms from the mapper. The output from the mapper could be decomposed and rearranged into sub-assemblies in the OOPM. Also, more decomposition methods need to be investigated.

8.6 References

1. Blaha, Michael R.; Premerlani, William J.; Rumbaugh, James E. Relational Database Design Using An Object-Oriented Methodology. Communications of the ACM. April 1988, pp.414-427.
2. Blake, Edwin; Cook, Steve. On Including Part Hierarchies in Object-Oriented Languages, with an Implementation in Smalltalk. European Conference on Object-Oriented Programming. June 1987, pp.41-50.
3. Borning, Alan. The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory. ACM Transactions on Programming Languages and Systems, Volume 3(4), October 1981, pp. 353-387.
4. Booch, Grady. Object-Oriented Design. Benjamin Cummings Publishers. New York. 1990.
5. Booch, Grady. Object-Oriented Development. IEEE Transactions on Software Engineering, Volume SE-12(2), February 1986, pp.211-221.
6. Booch, Grady. Tutorial on Object-Oriented Design. Conference on Object-Oriented Programming Systems, Languages, and Applications. October 2-6, 1989.
7. Coad, Peter; Yourdon, Edward. Object-Oriented Analysis. Prentice-Hall, Inc. 1990.
8. Constantine, Larry L.; Yourdon, Edward. Structured Design. Yourdon Press. 1978.

9. Cox, Brad J. Object-Oriented Programming, An Evolutionary Approach. Addison-Wesley, Inc. 1987.
10. Fulton, R.E. and Chao-pin Yeh. Managing Engineering Design Information. AIAA/AHS/ASEE Aircraft Design Systems and Operations Conference. Atlanta, Georgia. September 7-9, 1988.
11. Goldberg, A.; Robson, D. Smalltalk80: The Language and Its Implementation. Addison-Wesley, 1983.
12. ICAD. ICAD User Guide. 1989.
13. Imamura, S.; Kojima, T.; Sekiguchi, H. A Study on the Object-Oriented Product Model - Representation of Geometry and Dimension. Annals of the CIRP. Volume 37(1) 1988, pp.127-130.
14. Jacobsen, Ivar. Language Support for Changeable Large Real Time Systems. Proceedings of Object-Oriented Programming Systems, Languages, and Applications. September 1986, pp. 377-384.
15. Kolb, Mark. A Flexible Computer Aid for Conceptual Design Based on Constraint Propagation and Component Modelling. Presentation of the Doctoral Thesis Defense at Massachusetts Institute of Technology. October 30, 1989.
16. Loomis, M.E.S.; Shah, A.V.; Rumbaugh, J.E. An Object Modeling Technique For Conceptual Design. European Conference on Object-Oriented Programming. June 1987, pp. 192-202.
17. Meyer, Bertrand. Object-Oriented Software Construction. Prentice-Hall, Inc. 1988.
18. Minsky, Naftaly H.; Rozenshtein David. A Law-Based Approach to Object-Oriented Programming. Proceedings of Object-Oriented Programming Systems, Languages, and Applications 1987. October 4-7, 1987, pp.482-493.
19. Parametric Technology Corporation. PRO/Engineer User Guide, Release 3.0, Revision 3.
20. Pun, Raymond. A Decision Framework for Engineering Design. Masters Graduate Thesis. Georgia Institute of Technology. May 1990.
21. Rinderle, J.R. Implications of Function-Form-Fabrication Relations on Design Decomposition Strategies. Computers in Engineering, 1986. American Society of Mechanical Engineers. Chicago. 1986, pp.193-198.

22. Rogers, James. DeMAID: User's Guide to Design Manager's Aid for Intelligent Decomposition. NASA Langley Research Center. Hampton, Virginia. March 1989.
23. Shlaer, Sally; Mellor, Stephen J. Object-Oriented Systems Analysis, Modeling the World in Data. Prentice-Hall, Inc. 1988.
24. Steward, Donald V. Systems Analysis and Management. Petrocelli Books, Inc. New York. 1981.
25. Structural Dynamics Research Corporation. SDRC Geomod Solid Modeler. Version 5.0. 1990.
26. Suh, Nam P. Basic Concepts in Design for Producibility. Annals of the CIRP. Volume 37. Number 2. 1988.
27. Ten Dyke, R.P.; Kunz J.C. Object-Oriented Programming. IBM Systems Journal. Volume 28(3), pp.465-478, 1989.
28. Thomas, Dave. In Search of an Object-Oriented Development Process. Journal of Object-Oriented Programming. May/June 1989, pp.60-63.
29. Tomiyama, Tetsuo; Kiriya, Takashia; Takeda, Hideaki; Xue, Deye; Yoshikawa Hiroyuki. Metamodel: A Key to Intelligent CAD Systems. Research in Engineering Design. Springer-Verlag, Inc. New York. 1989, pp.19-34.
30. Wasserman, Anthony I. Tutorial on Object-Oriented Structured Design. Presented at the SCOOP East Conference. May, 1990. Tyngsboro, Massachusetts.
31. Wegner, Peter. Dimensions of Object-Based Language Design. Proceedings of Object-Oriented Programming Systems, Languages, and Applications 1987. October 4-7, 1987, pp.168-182.
32. Yoshikawa, H. General Design Theory and a CAD System. Man-Machine Communications in CAD\CAM. North Holland Publishing Company. 1981, pp.35-58.

Chapter 9 MULTILEVEL DESIGN APPLICATION

The hierarchical design approach, together with the object oriented model, needs to be explored in terms of growing levels of detail in a design. Chapter eight shows a single example of overall vehicle design where the major parameters were at the same gross level of definition. This chapter investigates the approach within the context of several levels of detail as a design refinement is presented. In particular, it applies the approach for design decisions related to a system (car), subsystem (door), and component (door crash stiffener). This allows one to proceed down the hierarchy to three levels and to investigate the multilevel linkages of the design variable linkages, constraints, and procedures. The application is largely geometry oriented and provides the opportunity to explore multilevel geometric linkages and the suitability of using commercial CAD systems to support such an approach.

9.1 Description of the Automobile Door Design Example.

The three-level problem, corresponding to the design of an automobile door, is shown schematically in Figure 9.1. Decision elements corresponding to the vehicle, door, and a crash stiffener structure located within the door have been identified. These decision elements correspond naturally to elements of the vehicle hierarchy. It should be noted that this is not the only possible decomposition of the problem into decision elements.

The system integration approach to design involves modelling the door at some level of detail, for example, overall geometrical outline (shape), weight, and side impact deformation. We then postulate some simple engineering theories relating these quantities, at the system level, to each other, and to other system parameters. System-level optimization results in values for the door/vehicle system interface parameters. These parameter values then enter the door design problem as constraints.

In a multiobjective problem, the door (subsystem design) problem may have local objectives which are considered independently from the vehicle system design. More typically, a cumulative constraint function is formed and used as an objective function for the subsystem design problem.

Up to this point, the flow of design requirements in this process has been strictly top-down. If feasible solutions to the subsystem design problem cannot be found, the multilevel approach provides for iteration with the system level problem. This is done by incorporating optimal sensitivity derivatives of the cumulative constraint function as constraints in the system-level optimization.

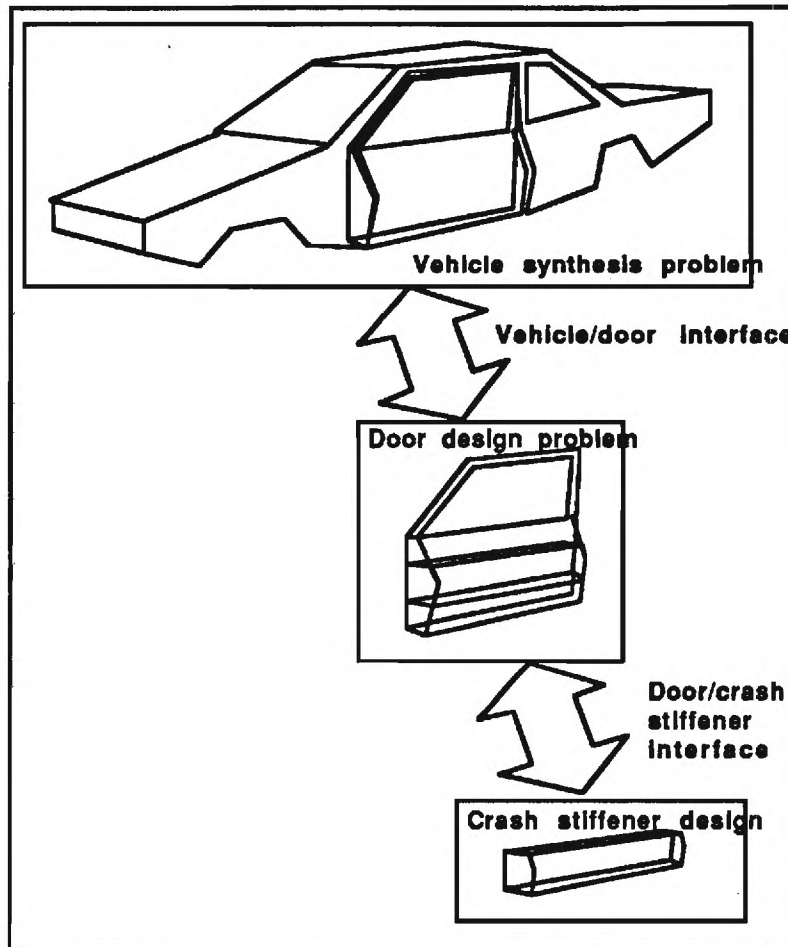


Figure 9.1. Hierarchical Design Problem: Automobile Door.

9.2 Software Infrastructure

An effective design environment framework is composed of five key components: a database, an expert system, a modeler, a set of analysis and optimization tools, and a CAD system. Each piece plays a distinctive role in the design process. For example, an expert system in conjunction with a database performs a functional decomposition of an object. The object is redefined into a finite number of forms. The object's function decomposes into a finite number of sub-functions. With a modeler, a set of feasible form-to-function sets are optimized using its user defined rules. These subsets support the overall function of the initial object. Once reaching an optimal solution, the database is updated and the object is processed through a CAD system for production and operation/support analysis.

Selection of suitable software utilized the following criteria:

- Flexibility - supporting all phases of concurrent engineering
- Hardware Independent - software compatibility on a variety of platforms
- Open Architecture - easy assimilation into existing engineering environment
- Industry Standard - support existing standards in graphics, networking, windows, and data exchange.

Additionally, the overall framework must create an environment which compliments the goals of concurrent engineering. It must lend itself to hierarchical decomposition. It must maintain a database which is manageable and prevents repeating past mistakes. Lastly, it must act as a catalyst for involvement by all members of the design team.

9.2.1 Software Platform

Selection of software platforms considered to various degrees existing industry standards. Micro-Station, EMS, CATIA, and CV were just a few of the types of platforms considered for this research. However, the two platforms selected were I-DEAS and ICAD. Each of these platforms possessed most or all of the five essential components of the software design framework. I-DEAS was principally selected for its feature-based capabilities. It is currently used for the system description and the finite-element analysis of the components. ICAD was selected for its parametric-based capabilities. It is currently used at the subsystem and component level. With the principle focus at the subsystem level, the research takes advantage of ICAD's ability to deliver deterministic, probabilistic, quantitative and nonquantitative information. Both of these platforms are addressed in greater detail below.

The current platforms require two databases. The principal database, Oracle, is a library for all design work and coordinates and facilitates operations in and out of ICAD. Oracle uses CLIPS to perform many of its export functions. I-DEAS uses the Pearl database for its operations. Discussions about the two databases and their functions are discussed below.

9.1.1.5 Description and Functions of Software Components

CLIPS: CLIPS is an expert systems facility. CLIPS, although written in C, looks and behaves like Lisp. Programming in the CLIPS expert system functionally decomposes a design objective into a number of feasible form-function sets. As expressed by Oracle data structures, forms and functions are described and linked to together. Each form "knows" what functions it can perform and within what boundaries it can perform those

functions. Each form also "knows" what other functions it requires to operate. Hence, in the initial stage, original objective functions are interrogated by CLIPS using forms and functional capabilities described in Oracle. This process results in a series of feasible form-function sets which are modelled to determine optimal form to function performance.

ORACLE: Oracle provides three principle services to the design environment. It serves as a communications hub between other components. Oracle provides a global model definition that each team member and each software product can share. Hence cooperation is easy to maintain and design histories as well as decision paths are documented. Finally, Oracle provides the general reference information needed for the design.

The design environment's databases provide crucial linkages between each of other components. With a number of products working together on a design, creating one to one translators, for each possible connections, is an enormous task. The task increases factorially with the number of products. Since expendability is important to a flexible software framework, a central database is used which requires only one pathway per product. Of particular interest is the I-DEAS pathway to ICAD. Developing a robust capability to translate model information between the Pearl database in I-DEAS and Oracle paves the way for system, subsystem, and component interactions.

A globally shared model description is contained in Oracle. Once the function based decomposition is completed, Oracle communicates each of the proposed concepts to the modelers. Each modeler exercises its design rules on the set of forms composing the proposed concept. The modelers determine how best to build the proposed set of forms into a functioning design model, relying on Oracle for coordination. Oracle coordinates the contributions of each team member and maintains design history and design decisions.

Oracle are also serve as an online library for reference information. Available parts, previous designs, costs, and material attributes are all be maintained in one place so that work is not duplicated and a more self-contained design history is maintained.

ICAD: ICAD was chosen as the subsystem and component modeler because it offers the most capable environment for complete parametric design. ICAD's parametric programming results in an interactive application environment, the designer is led through a series of questions which prompts the designer for inputs. The application creates geometric and property outputs based on rules and design data imbedded in the application's code. Updates to this environment, particularly the user interface development and database integration are easily assimilated into the platform with little downtime.

Where I-DEAS utilizes feature-based design, ICAD builds designs with IDL, which is ICAD's design language. There are three important advantages to programming in this design language. Firstly, the process offers total control over all aspects of the design model. Secondly, the designer is not constrained by a small number geometric primitives. Thirdly, ICAD gives the coder the flexibility to interface with other platforms or write code directly into the environment.

9.3 Multilevel Model of Door Geometry

The software infrastructure is used to capture the multilevel model of the door and crash bar consistent with system (car) level geometric constraints. The following summarizes selected information describing the door. The door subsystem was modelled with ICAD, and the detail of the crash bar and other components are modelled and analyzed with IDEAS. The relationship of the design to the overall vehicle measures of Please and Protect are also discussed.

9.3.1 Door Subsystem Modelled with ICAD

A door design model in ICAD satisfies the design requirements for functionality, realistic detail, and parametric simplicity. Two functions of pleasing and protecting affect the door's form. The doors created in response to these functions must contain sufficient detail to be realistic while remaining parametrically simple. For example Figure 9.2 depicts an appealing door created from less than twenty parameters.

Functional modelling criteria are applied to the subsystem attributes and the component forms. At the subsystem level, the function please is fulfilled by the door's geometry, window area, and weight distribution. All are subsystem attributes. Within the door's subsystems, the function protect is fulfilled by a stiffener, latches, hinges, and frame members. Protection is provided by these component level forms. All are primarily dependent on an input called Impact, representing crash intensity.

To insure realistic detail and protection, the door must fit into the car. Thus system level geometric constraint dictates many of the door's ICAD inputs. As seen in Figures 9.3 and 9.4 which indicate door inputs, this geometric constraint is sufficiently detailed to insure compatibility between the system and the subsystem. The ICAD code creates the geometric components based on user-defined inputs and design rules.

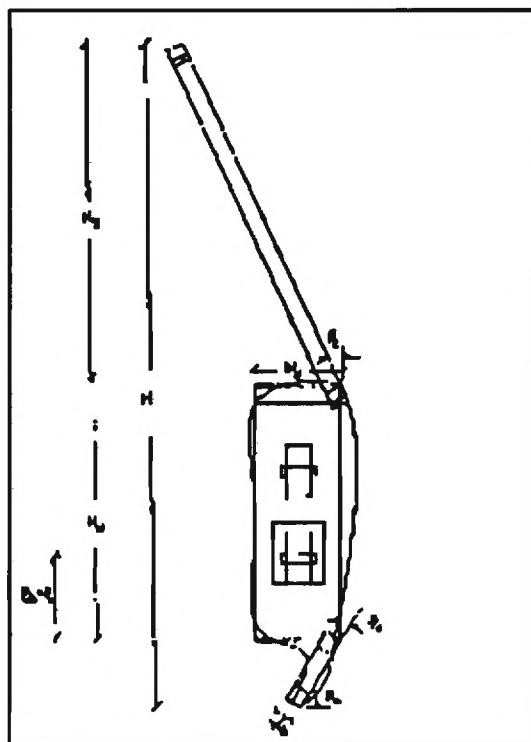


Figure 9.4. Door Inputs (Longitudinal View)

Design rules for the door fall into two categories: those which insure geometric compatibility between the door components and those which determine the size, placement, and number of each component based on pleasure and protection. A pseudo-code translation of many of these rules is included below in Table 9.1.

Rules governing geometry are compatibility relations such as:

- *The lower body section's front frame's length is the distance up the height of the lower body section and in β_b degrees.*
- *The middle body section's front, right, and bottom corner must coincide with the lower body section's front, upper, and right corner.*

Example IDL Code:

```
(middle-body-f :type box
  :Display-controls '(:color :blue)
  :width (the :wd)
  :height (the :hd))
```

```

:length (the :td)
:orientation (:numeric (roll :longitudinal (degree (+ (the
:Bb)
(the :Bd) -90))))
:position-about
(:local-point (the :middle-body-f (:vertex :right
:bottom :front))
:model-point (the :lower-body-f (:vertex :right :top
:front))))

```

[All frame members are sized and positioned in this manner.]

- *The stiffener is position within the middle body section. Its vertical placement is a fraction of the middle body section's height. planes of the door's total structure and depicts the skin of the door.*

Rules implementing functional requirements are relations such as:

- *The number of latches is a function of impact.*

$$\text{number} = \text{Impact} / 500 \text{ rounded to the closest integer}$$
- *The latches are to be evenly distributed along the rear middle body section frame.*
- *The size of the latches is a function of door geometry, number of latches, and the impact power.*

*Latches are 40% the width of the middle body section
Length is the thickness of the middle body section
Height is $\text{Impact} / (1000 * \text{number})$*

- *The number and placement of hinges is a function of impact and pivot axis.*

If standard then $\text{number} = \text{Impact} / 500$ on the front of the middle body section

If gull wing then $\text{number} = \text{Impact} / 300$ on the top of the top window frame

If swing up then $\text{number} = 1$ on the inside, front, and top of the middle body section.

- *The size of the hinges is dependent on the impact, axis of pivot, the geometry of the door, and the number of hinges.*

*If standard: $\text{radius} = \text{Impact} * \text{length} * .000002$
 $\text{length} = \text{Impact} * \text{number} * \text{door section height} * .0002$*

*If gull wing: $\text{radius} = \text{Impact} * \text{height} * .000002$
 $\text{length} = \text{Impact} * \text{number} * \text{door section height} * .0002$*

*If swing up: $\text{radius} = \text{Impact} * .000003$
 $\text{length} = \text{door section width} * .03$*

- *The sizes of the window frames vary based on pivot axis, impact, and door geometry.*

<p><i>If standard or swing up then window frame height and width are .1</i></p> <p><i>If gull wing then window frame height = Impact * .00006 * length * .35 and frame window frame width = Impact * .00006 * length * .5</i></p> <p>• <i>The size of the stiffener is a function of door geometry and impact intensity.</i></p> <p><i>Length is determined by the position within the middle body section</i></p> <p><i>Width is 60% of the middle body section's width</i></p> <p><i>Height = .0001 * length * Impact</i></p>

Table 9.1. Pseudo-Code for Subsystem Level Door Description

Example IDL code that is used to implement the pseudo-code is shown below in Table 9.2. Each of the rules in Table 9.2 is designed to produce reasonable results and reflect the contribution of the subsystem's function, however, for the purposes of this study, no attempt was made to be exactly correct in terms of engineering theories and models.

<pre> (defun stiffener-height (impact length) (* length .1 impact .001)) (stiffener :type box :Display-controls '(:color :red) :width (the :wds) :height (the :hds) :length (the :lds) :orientation (:numeric (roll :longitudinal (degree (+ (the :Bb) (the :Bd) -90)))) :position-about (:local-point (:face-center :front) :model-point (mapcar '+ (the :door-section-f (:edge-center :front :bottom)) (list 0 0 (* (the :hd) (the :pds)))))) </pre>

Table 9.2. Sample IDL Code

Interaction between system, subsystem, and component models can be seen both within the ICAD models and between the ICAD and the I-DEAS environments.

From the system level, the car model in I-DEAS specifies the basic geometry of the subsystem door model in ICAD. At the subsystem level the door may need to open gull winging. This may be due to a system level, height or pleasure requirement. Hinge arrangement and frame sizes must then change at the subsystem level. Figures 9.5, 9.6, and 9.7 depict these

changes. If at the subsystem level gull-wing doors are chosen, then at the system level the roof of the car must be designed to accommodate the new loads and make room for hinges.

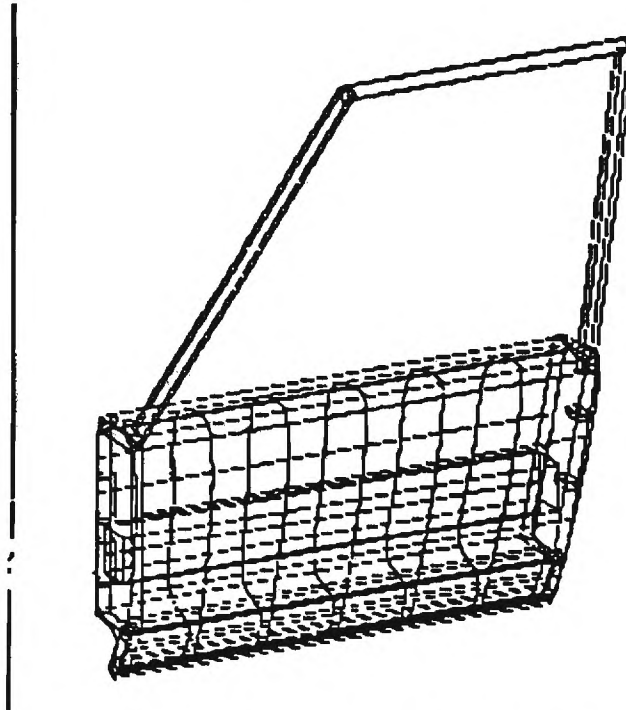


Figure 9.5. Example of Changes In Door Configuration Showing Selection of Standard Hinges.

A second decision path which satisfies the protection function is highlighted below. To protect, the door must endure some impact without harming the passenger. This impact criteria directly effects the door's stiffener. It also affects the door's frames, latches, and hinges. With I-DEAS's FEM, finite element analysis is performed on the component model of the stiffener. Results from that component design are reflected in the ICAD subsystem model of the door. As the door's geometry changes, differences in the number, size, and placement of hinges and latches occur. These differences must be fed up to the system model of the car in I-DEAS to insure that the door can still close safely.

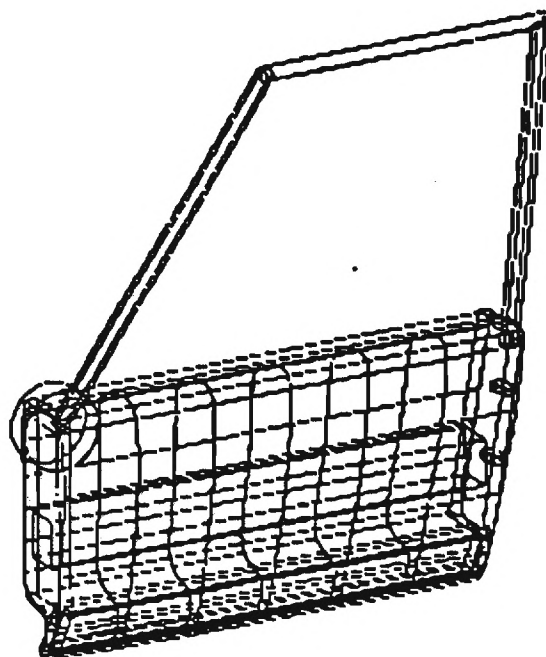


Figure 9.6. Example of Changes In Door Configuration Showing Selection of Swing-up Hinges.

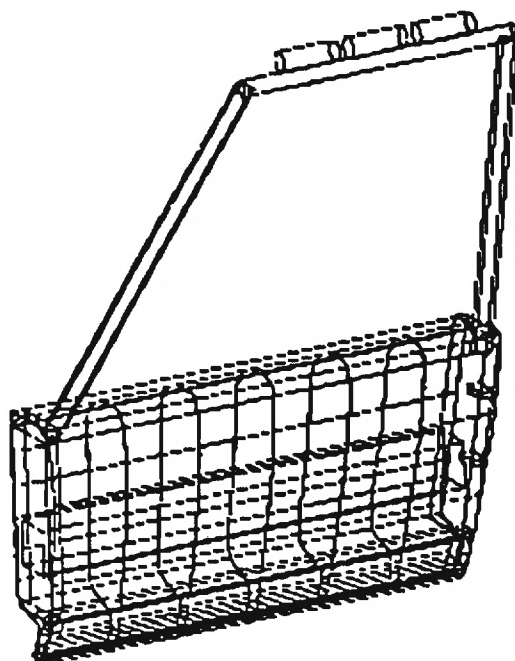


Figure 9.7. Example of Changes In Door Configuration Showing Selection of Gull-wing Hinges.

9.3.2 Detail Component Modelled with IDEAS

IDEAS was selected as a suitable featured-based program since it possessed four of the five software components of the design framework. While it lacks an expert system, I-DEAS is a fairly comprehensive CAD environment which is suitable to describe the system. The salient advantage of the I-DEAS package is its feature-based capabilities and this, more than any other reason, motivated its selection. I-DEAS processes geometric data to the Pearl database, as shown in Figure 9.8.

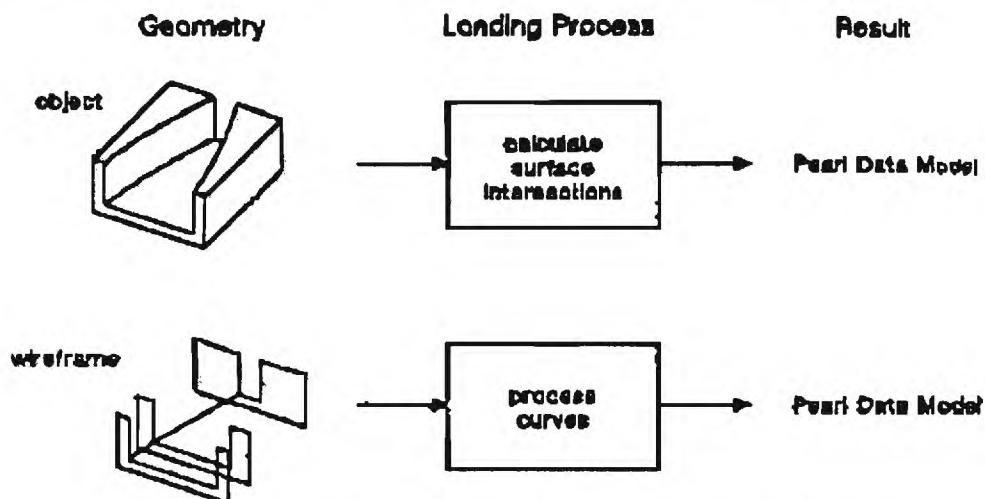


Figure 9.8. Loading Geometry Into the Pearl Database

The source of its data is either a wireframe or an object. The database contains tables that define and describe the geometric data. A set of tables is referred to as a relational data model. After the geometry is loaded into the database, newer versions of an object or wireframe can be added while older versions can be deleted. The Pearl database interfaces with other databases by accessing an administrator, which is a library of interface subroutines.

Version 5.0 of I-DEAS has been recently released and is unique from earlier versions in that it allows for featured-based definitions. This is a significant advancement over traditional solid object modeling methods. An advantage of user-defined features is the modeling of objects which possess variable definitions. One example is hinge placement on the frame of a car. By changing the variable dimensions, the dependent dimensions of the feature are automatically adjusted. So the dimensions which shape the hinge are dependent upon variables defined by the user.

In the example provided in Figure 9.9, the user has feature defined a plate with a slot hole. Whenever this feature is accessed, the dimensions are of fixed value except for the plate length. Each time the user specifies the length the object is automatically generated. I-DEAS stores features in a library which allows other users to share access to the design.

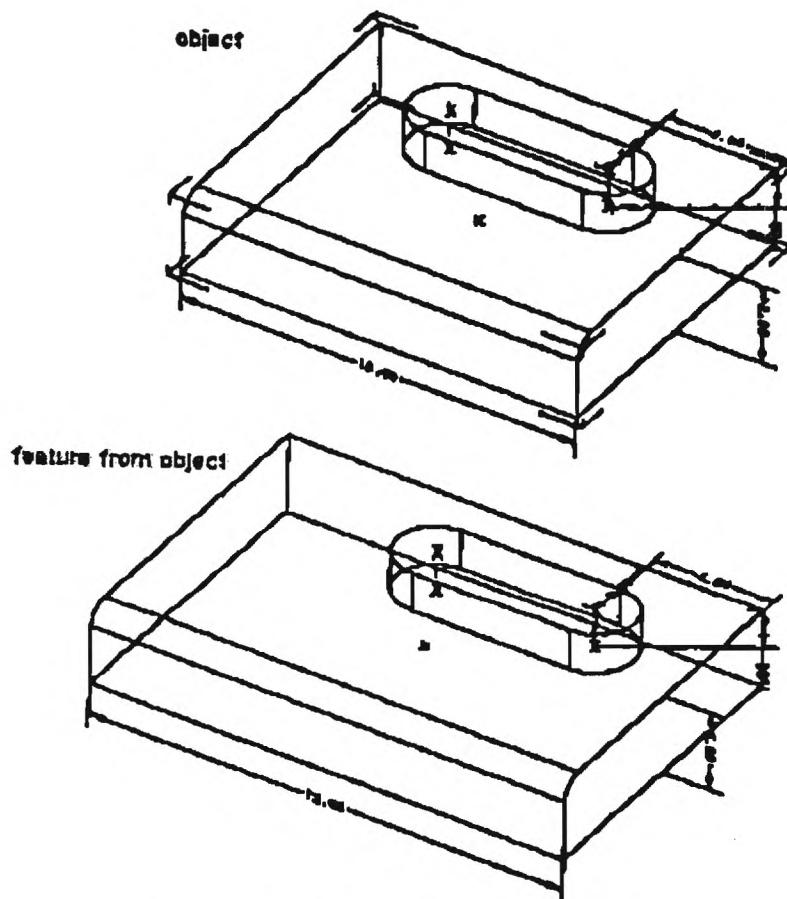


Figure 9.9. Example of a Feature.

Feature-based modeling controls both the dimensions and the orientation of the entities through parameters. Each time a feature is accessed, its parameters are processed so that the user can input new values for the parameters. The feature's history is re-processed with the new values to modify the geometry of the feature. The feature's parameters allow members of the design team the capability to set design standards and controls. Hence a feature-based design at the system level can interface with form variations produced at the subsystem level. A parametric-based form defined in ICAD can be fed through the Oracle database to the Pearl database and finally, into the feature-based system in I-DEAS.

To further explain the purpose of feature parameters consider the example in Figure 9.10 below. The feature is a block with a circular hole cut through it. Note that the feature consists of two entities: the block and the hole. The user can create parameters which control either the block and/or the circular hole. The user may choose to control the hole by creating a parameter that controls the translation of the hole along a given axis. In

the example above, the hole is the controlled entity. The translation of that controlled entity along the x-axis is the controlled detail. Here the specified parameter affects only the controlled detail of the control entity. This feature allows the user to move the hole along the x-axis each time it is accessed.

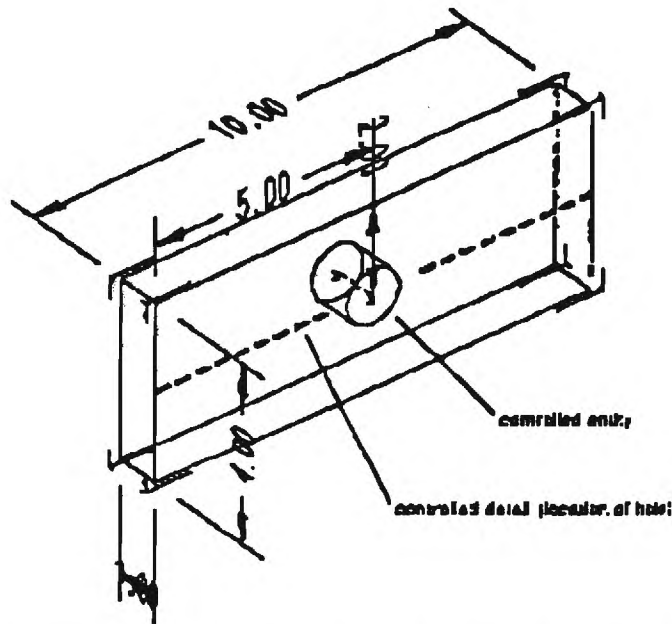


Figure 9.10. Example of a Feature Parameter

In addition to user-prompted inputs, feature parameters can also be expressed equatorially. An equatorial parameter defines a mathematical relationship between two other parameters. Some examples of equations are:

- $\text{second} + \text{sqrt}(\text{first})$
(Calculates the value of "second" plus the root value of "first")
- $\text{cos}(\text{theta})$
(Calculates the cosine value of "theta.")

Finally, parameters may be limited by the user to a minimum/maximum value. Hence constraints at system level can guard against any inadmissible form which is generated by the subsystem level.

Analyzing the door's function to protect requires predicting the behavior of its component structure, which in this case is a beam. The beam's behavior is characterized by its displacement. Therefore, beam displacement is analyzed in I-DEAS utilizing finite element tasks.

The finite element method estimates stress at each node. A separate estimate is calculated from each element connected to each node.

Displacement results are stored to analysis datasets as vector data at nodes. Stresses are written to analysis datasets as symmetric tensor data at nodes on elements. Figure 9.11 depicts the six stress values calculated for the beam elements.

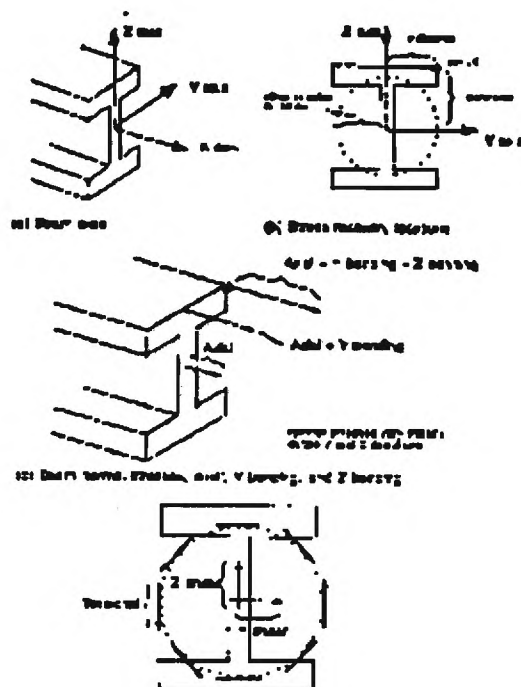


Figure 9.11. Finite Element Analysis of Component

Finite element analysis is also applied to door components to assess their performance under load. Applying a parabolic tetrahedron mesh to an object increases the precision of the analysis. Figure 9.12 depicts the results of a stress analysis on a door hinge tap constraints at the support. The dashed lines indicate the original geometry.

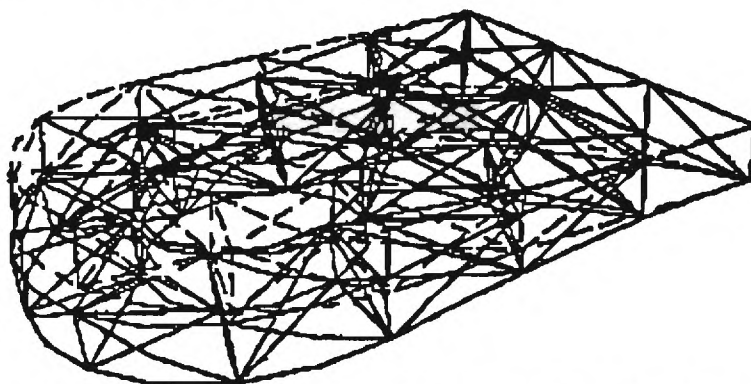


Figure 9.12. Analysis of Deformation Results

9.4 Summary

This chapter has presented an example of a car door to illustrate and test the multilevel hierarchical design approach and the adequacy of commercial software to provide a software infrastructure. The results are encouraging and indicate that complex geometries and detailed analyses can be incorporated in the method. The results also illustrate how linkages can be made between a detail component design and the overall vehicle performance measures such as Please and Protect. The results also show some of the limitations and advantages of the software and the methodologies.

Chapter 10 SUMMARY OF ACCOMPLISHMENTS AND ISSUES

This report summarizes the work to date in the areas of developing the requirements for the Object Oriented Vehicle Model and for Hierarchical Decomposition Design Methodologies.

The work in the OOVm has developed the preliminary decompositions in the function and form and has started to relate them to the Voice of the Customer and the Baseline Parameters. A set of inputs and outputs has been determined for the model and subsets defined for testing. The model has been tested against previous GM work to assure that nothing has been left out, as well as that compatibility is maintained with GM terminology. A first attempt at integrating the model with a database, an expert system, and the Voice of the Customer has been successfully completed.

In the area of hierarchical decomposition, the work summarized in this report represents the initial efforts to clearly define the research problem and to formulate suitable and effective approaches for dealing with it. The report briefly outlines the work completed to develop a prototype design problem and to implement some of the basic software tools necessary for its solution. Selected examples are explored to clarify issues, identify potential benefits, and uncover problem areas in a hierarchically structured design environment.

In summary, the two activities carried out to date in these studies have resulted in the following accomplishments:

1. Development of an overall design strategy based on hierarchical design methods and object oriented modelling for addressing automotive conceptual design driven by customer requirements.
2. Definition of the vehicle from the customer perspective via QFD.
3. Definition of the vehicle from an operational perspective in terms of vehicle functions.
4. Definition of the vehicle from a physical perspective in terms of form descriptions.
5. Identification of sample linkages among the perspectives.
6. Development of a theoretical framework for multilevel automotive design.
7. Selected experimental applications to test the design approach concepts and possible software infrastructures for both overall design and multilevel component design.

The basic concepts for the proposed design strategy have been conceived, discussed, characterized, and examined. It is felt that the design strategy provides a good foundation on which to build. It contains many highly desirable features of a new framework for design, such as

1. modularizes the process and models,
2. provides direct customer requirements impact on design,
3. utilizes advanced information management concepts to benefit an information driven process,
4. exploits evolving software products based on object oriented concepts,
5. facilitates the use of computing technology to aid design bookkeeping and decision making,
6. incorporates effective use of multilevel optimization concepts, and
7. incorporates multiview approaches to design.

While the above design approach has good concepts and high potential, there are certainly many issues which remain. Some of these include:

1. Refinement of concepts to further levels of definition,
2. Actual linking of vehicle multiple view,
3. Incorporation of realistic design variables,
4. Representation of actual design experiences,
5. Integration of design rules,
6. Better integration of the design information, process, views, and multiple levels, and
7. Scaling of multiple optimization functions.

It is believed that such issues can be addressed within the context of application of the model and methods to realistic design scenarios and implementation of the approach in prototype software. Plans for addressing these issues are discussed in more detail in chapter 11.

Chapter 11 PLANS

11.1 Planned Tasks

The next year's work will address the four key tasks outlined below and their respective areas of emphasis relative to these tasks. This work focuses on implementing the results of this work in software on appropriate hardware platforms. The Object Oriented Vehicle Model will be expanded and tested within the framework of the hierarchical decomposition strategies.

In particular, the proposed second year effort will study the applicability and effectiveness of three general decision support methods:

- deterministic approaches such as the DSS method,
- probabilistic approaches, and
- methods based on multi-level optimization.

The general outlines of these approaches have been presented in the present report. Each will be examined in greater detail, and their suitability for use in automotive design will be studied in the context of the door design problem described in this report.

A key part of the proposed work will be to continue development of the prototype software system initiated during the first year. While most of the major software components have been identified and preliminary work has been initiated to implement the prototype problem, many of the details have not been adequately resolved. Work during the second year will be focused on completing a reliable working prototype software system. Particular attention will be directed to the following key areas:

- Development of the subsystem representation where it is anticipated that much of the decision-support methods will be utilized,
- Development and implementation of suitable information framework tools to allow ready and effective interaction between the system, subsystem and component levels.
- Implementation of decision-making processes at each of the three hierarchical levels and study of the resulting system-wide interactions.

11.1.1 Task 1- Implement and Test Prototype Software Framework

In this task, a software framework will be implemented and tested for the automotive conceptual design process based on an Object Oriented Vehicle Model using the Hierarchical Decomposition Approach. The software framework will include appropriate user interface, information management, optimization and expert system capability integrated on a

various hardware platforms. The framework will include storage, management, and linkage of information characterizing the vehicle at several levels including both functional and form descriptions and at several levels of detail. The framework will be tested on selected conceptual design examples at the gross vehicle level and at a component level. The key goal of this task is to verify the completeness and linkages of the software framework and to ensure that management of the vehicle form, function and design rules is consistently carried out.

11.1.2 Task 2- Refine and Extend Automotive Design Methodology

This task will build on the studies to date in design methodology. The methodology has included consideration of quality function deployment, function and form relationships and their respective hierarchical decompositions in describing a vehicle design. Each of these relationships provide different representatives of an automotive design. Work to date has provided representative descriptions of the vehicle in terms of the three relationships and illustrated sample linkages. An initial set of relationships has been defined in each of these areas. This task will expand in the level of hierarchical decomposition of the vehicle description and will develop ways to link the various decompositions. This task will be to establish a clean thread of linkage between the high level customer requirements associated with QFD and the form description of the vehicle and to define how optimization concepts can be applied to achieve "best" designs.

11.1.3 Task 3- Develop a Database Approach for Automotive Design

It is critically important to establish an approach to managing the information associated with vehicle design. As a vehicle design is continually refined in terms of customer requirements, functional requirements or form description, the qualities of information escalate rapidly. Various approaches to database management exist including relational, object oriented or object concepts on a relational environment. Each of these have certain assets and liabilities. Relational approaches are more flexible, but can have performance difficulties. Object oriented methods can be a more natural way to decompose a large product but can be less flexible, and have poorly developed performance experience. Combinations which build on both may be more attractive at least until object oriented software matures. This task will explore the various approaches, recommend a strategy, and incorporate it in the software framework.

11.1.4 Task 4- Apply Prototype Software Framework to Representative Vehicle Design

The prototype software and methodology developed in the work to date and in the above ongoing tasks will be applied to the study of representative automotive design. In particular the study will investigate how changes in one of the three customer requirements such as Please, Protect or Transport can result in changes in vehicle component design. Such design changes can then subsequently lead to attendant modification in the other two customer requirements. The software infrastructure will provide an opportunity to carry out design tradeoffs and will show examples of linkage between high level customer requirements and component design the application will use one or more baseline vehicle configurations as test cases and will investigate sensitivity of the configuration to changes in customer requirements. Selected optimization concepts will be tested for test vehicle to show how various customers requirements can be best met.

11.2 Testing Methodology

The test philosophy is to take these and other scenarios and determine the effects of changes in the baseline parameters of a representative vehicle throughout the breadth of the vehicle as a method of studying the integrative ability of the model, design methodology and information storage mechanisms. It is not the object of this study to delve into detail design of components.

Testing will focus on representative design scenarios to test the model and the underlying design methodology. We intend to explore the functions Transport, Please and Protect using a car which has nominal baseline parameters. The following scenario can be envisioned to test the sensitivity of the functions to changes in each other. A requirement for the car is to Access Passenger Space in the Transport function. Increasing the size of the door increases the Transport function by increasing the value of Access Passenger Space. Increasing the door size also affects the Protect function, as the door would need to be strengthened to maintain a constant value of protection. If the door size is modified, the value of Transport may be reduced (the car would weigh more, and hence lose fuel economy), and the value of Please may be reduced because the heavier car doesn't perform as well. This may lead one to increase the size of the engine. One can easily see the complex network that quickly develops. The Object Oriented Vehicle Model, interacting with the Hierarchically Decomposed Design method (developed in the parallel contract), should allow one to more easily see the interactions of design decisions on the form and performance of the vehicle. A second scenario which can be imagined involves the Please Function. If a customer wants to increase the Entertain function, there are a multitude of changes that could occur. If the exhilaration of acceleration is of importance, the affect on the size of the car, its engine, and the Transport and Protect functions need to be explored. A third scenario

envisioned is that of packaging. The constraints of the size of the components and of the final car (wheelbase, length, width, height) affect the location and placement of these components. The ordering of their design and placement will be explored using the design methodology and hierarchical decomposition.